Energy-Efficiency Programming
Introduction

Dr. Manuel Dolz, Michael Kuhn, Dr. Julian Kunkel, Konstantinos Chasapis
Prof. Dr. Thomas Ludwig
Wissenschaftliches Rechnen
Wintersemester 2014/2015

Motivation

- High Performance Computing
  - Optimization of algorithms applied to solve scientific complex problems
- Technological advance ⇒ Performance improvement
  - More computing power and storage space
  - Multicore processors, accelerators and coprocessors
- HPC data centers ⇒ High energy consumption!
  - Growth of the Total Cost of Ownership (TCO)
  - Power wall towards exascale computing

Concurrency and efficiency

- Green500 vs Top500 (June 2014)

Rank Top/Green	Site	Technology	Performance (TFLOPS)	Power (MW)	Efficiency (MFLOPS/W)
1/49	**Tianhe-2** National University of Defense Technology	Intel Xeon E5 + Intel Xeon Phi	33,862.7	17.8	1,901.82
435/1	**TSUBAME-KFC** GSIC Center Tokyo Institute of Technology	Intel Xeon E5 + NVIDIA K20x	151.8	0.035	4,389.82

Concurrency and efficiency

- Green500 vs Top500 (June 2014)

Rank Top/Green	Site	Technology	Performance (TFLOPS)	Power (MW)	Efficiency (MFLOPS/W)	MW to EXAFLOPS?
1/49	**Tianhe-2** National University of Defense Technology	Intel Xeon E5 + Intel Xeon Phi	33,862.7	17.8	1,901.82	525.65
435/1	**TSUBAME-KFC** GSIC Center Tokyo Institute of Technology	Intel Xeon E5 + NVIDIA K20x	151.8	0.035	4,389.82	230.56

Most powerful nuclear reactor under construction in France: Flamanville (EDF, 2017 for 10 billion €) 1650 MW

Concurrency and efficiency

- Green500 vs Top500 (June 2014)

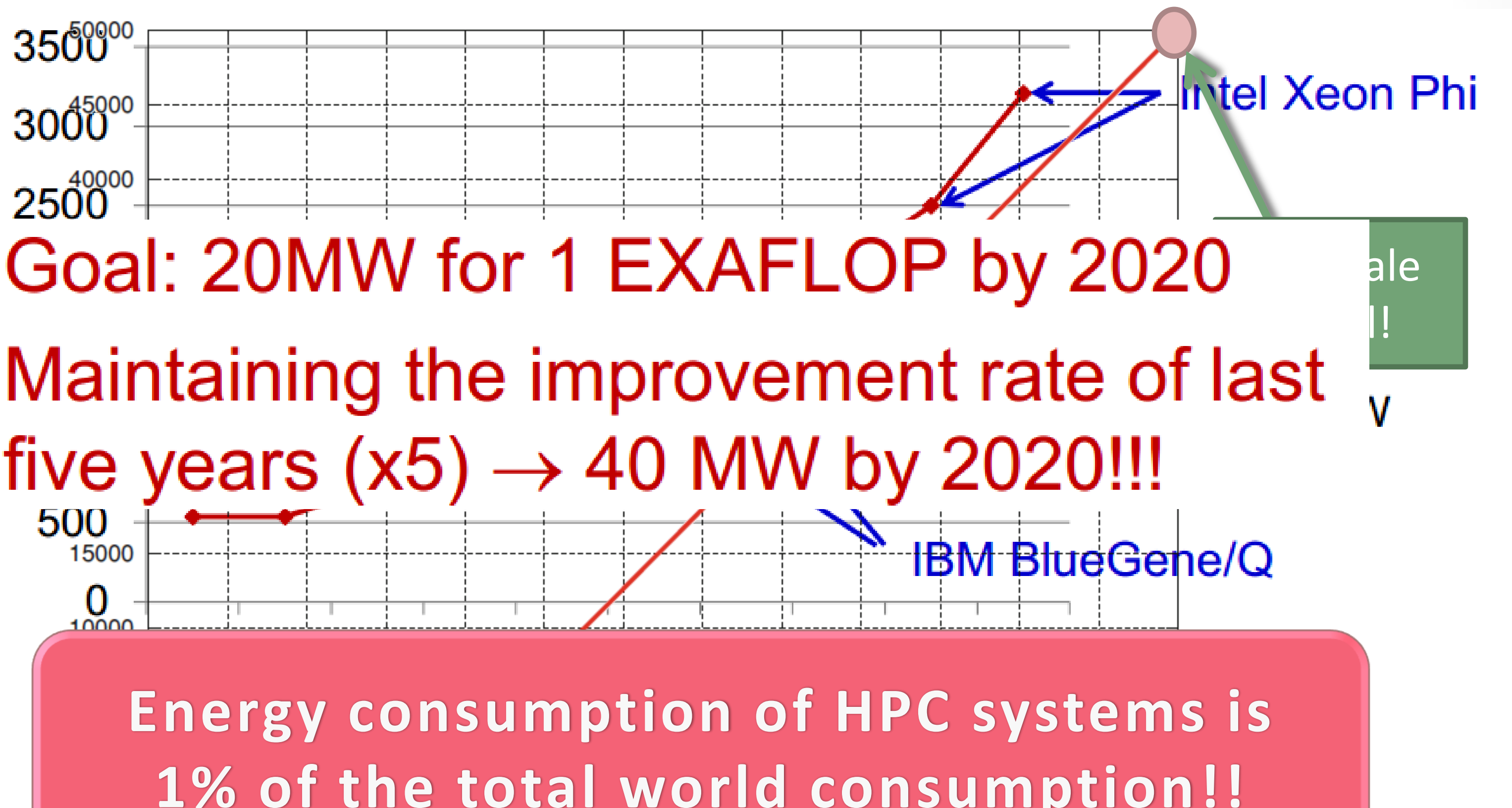
Rank Top/Green	Site	Technology	Performance (TFLOPS)	Power (MW)	Efficiency (MFLOPS/W)	EX
1/49	**Tianh** Na Ur De Te	Intel Xeon	33,862.7	17.8	1,901.82	525
435/1	**TS** GSIC Center Tokyo Institute of Technology	E5 + NVIDIA K20x				

1MW ≈ 1M€ !!

Most powerful nuclear reactor under construction in France: Flamanville (EDF, 2017 for 10 billion €) 1650 MW

Performance and efficiency trends

- Goal ⇒ Build and Exascale system (10^{18} FLOPS) without exceeding 20 MW



Goal: 20MW for 1 EXAFLOP by 2020

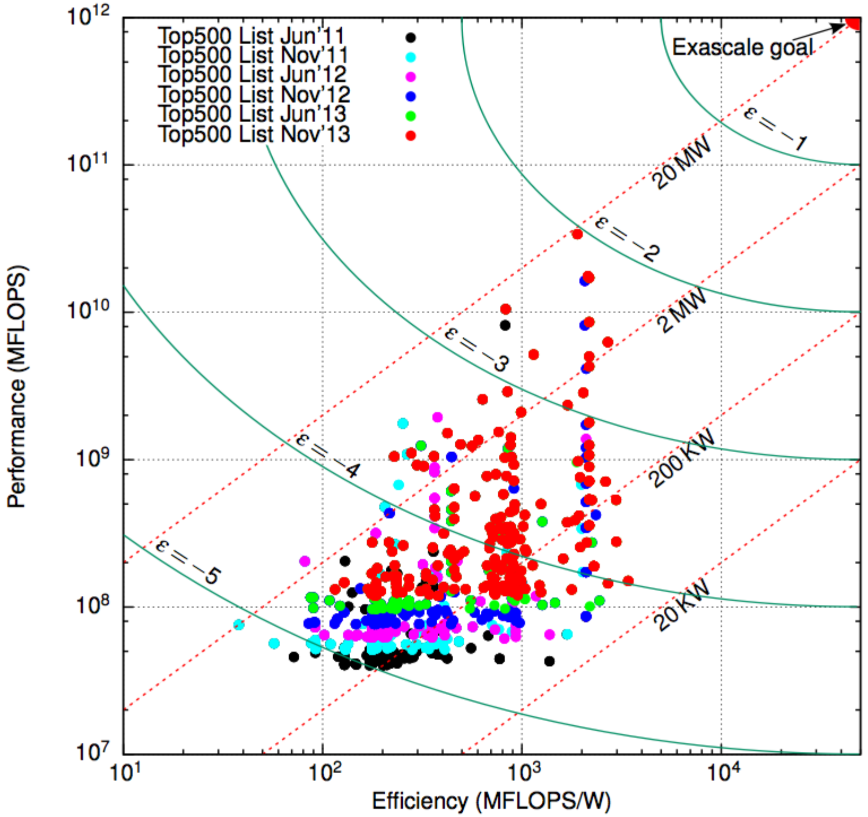
Maintaining the improvement rate of last five years (x5) → 40 MW by 2020!!!

Energy consumption of HPC systems is 1% of the total world consumption!!

Performance and efficiency trends

- Goal ⇒ Build and Exascale system (10^{18} FLOPS) without exceeding 20 MW

Performance-Efficiency scalar graph for the Top500 supercomputers from 2011 to 2013



Power trends of some supercomputers have almost reached the power wall being 100 away of the Exascale goal!

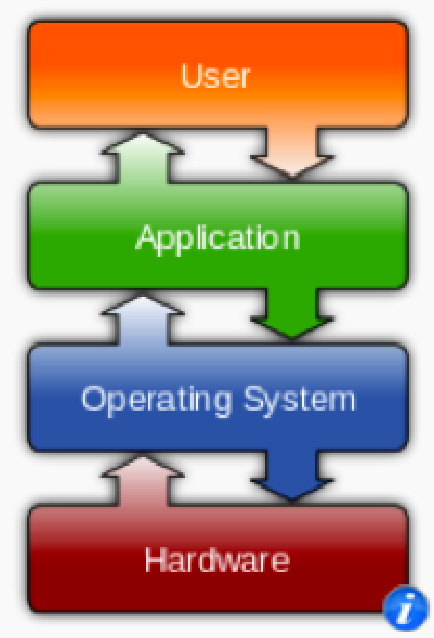
What we can do?

- Reduce energy consumption!
  - Costs over lifetime of an HPC facility often exceed acquisition costs
  - Carbon dioxide (CO_2) is a hazard for health and environment
  - Heat reduces hardware reliability
  - Scientific apps. are in general energy-oblivious
- Solutions?
  - Optimize applications from performance and energy!
    - **Use hardware features for power-saving mechanisms**

Energy-Efficient Programming

Energy Efficient Programming

- Methods for energy efficient programming at different levels:



- Approaches and concepts for **Energy Efficient Programming:**
  - Application software efficiency
  - Operating system optimizations
  - Common problems and solutions

Energy Efficient Programming

- Computational Efficiency (Performance) ➔ Get the work done as quickly as possible!

- Energy efficiency ➔ Minimizing energy used to complete a task!

- A task does not necessarily have to be completed in a shorter time
  - however, the computer can return sooner to a low power-state!

- Approaches to increase **Energy Efficiency**:
  - Use of efficient algorithms and data structures
  - Multi-threading
  - Efficient use of loops
  - Vectorizing and instructions sets
  - Efficient use of programming language
  - Energy efficient libraries and drivers

Use of efficient algorithms and data structures

- **Problem: Insensitive choice of algorithms and data structures may lead to siginificant energy wasting!**
- **A complete area of research in Computer Science!**
- The right choice of algorithms and data structures can make a massive difference in **software performance**!
- Therefore, energy efficient programming requires high performance algorithms!
  - Please, complete the work faster and go to sleep! ☺
- Example 1: **Sorting 200,000 double values**
  - ***Bubble sort:*** *O(n2)* - 10,800 Joules
  - ***Heap sort:*** *O(n log(n))* - 7325 Joules!

  33% less energy!
- Example 2: **Solving the Towers of Hanoi Puzzle** (in C++)
  - ***Iterative version:*** 1656.26 Joules
  - ***Recursive version:*** 322.22 Joules

  88% less energy!

But take care! Each single algorithm should be measured to find the greenest configuration!

Efficient use of loops

- **Problem: Careless programming of loops and overuse of spinning polling loops may lead important energy wastings!**

- **Efficient desgnt of loops:**
  - **Loop unrolling:** combine instructions called in multiple iter. Into a single one
  - **Avoid polling loops**: avoid to repeteadly check to see if a condition is true!
    - Use asyncronous/blocking methods
    - MPI/OpenMP: Use blocking waiting modes
    - Runtimes: Replace busy-waitings and use asynchronous waiting methods

Do nothing efficiently!

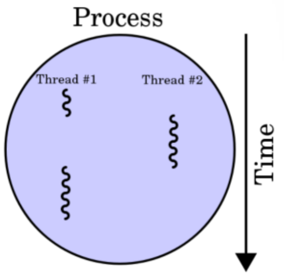
The hardware does not know about the program that is being executed!

Only if processes/threads are efficiently (idle) waiting , hardware can promote processors (cores) to energy-saving states!

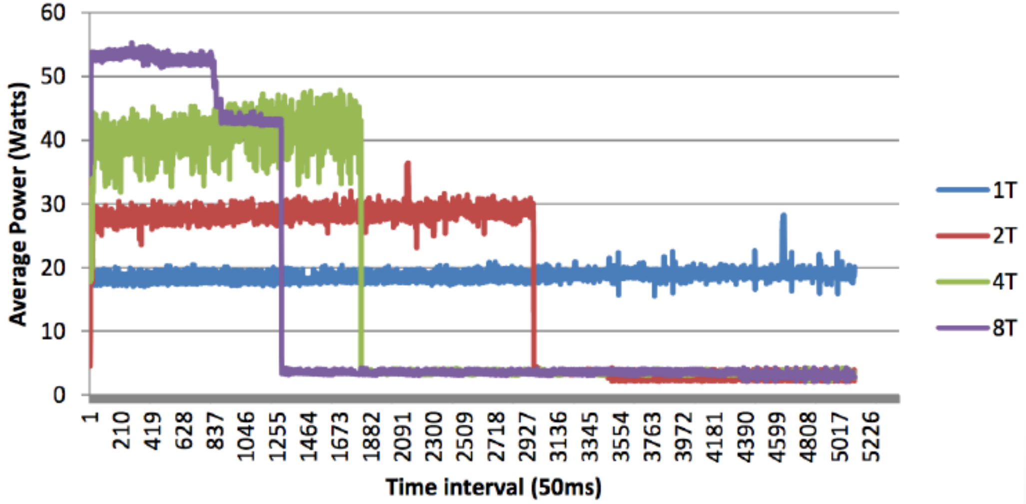
But take care: going to sleep for a very short period may not be efficient!

Multithreading

- **Problem: Single threaded applications are inefficient and waste energy**
- **Thread:** Smallest sequence of programmed instructions, inside a process, that can be managed by an operating system scheduler
  - *Implementations:* P-Threads, OpenMP, TBB, etc.
  - *Multithreaded programming:*
    - Parallel programming inside a process



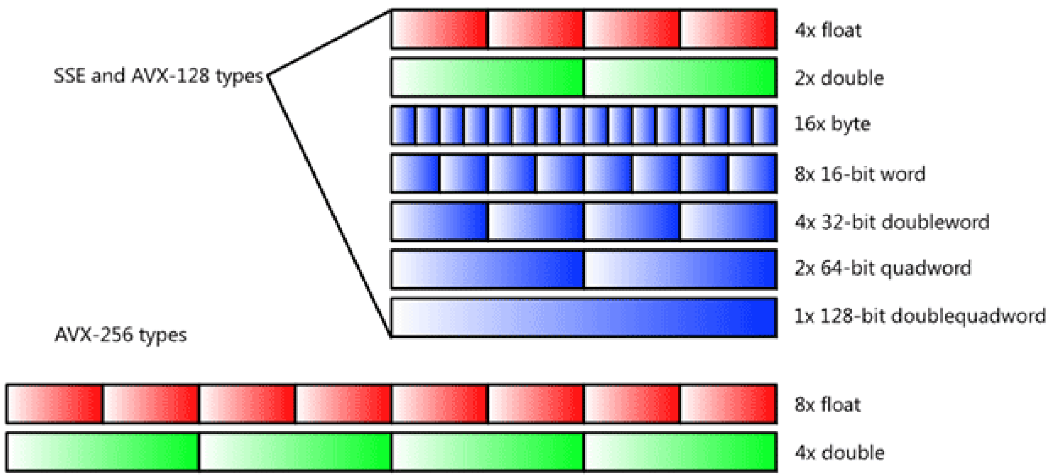
- **Technology advance**: Multi-core processors, Coprocessors, GPU, etc.
  - Doing an efficient use of the core's architecture we can go faster and save energy



Vectorization and Instruction Sets

- **Problem: Use of scalar code rather than vectorizing may lead to inefficient software and waste of energy**

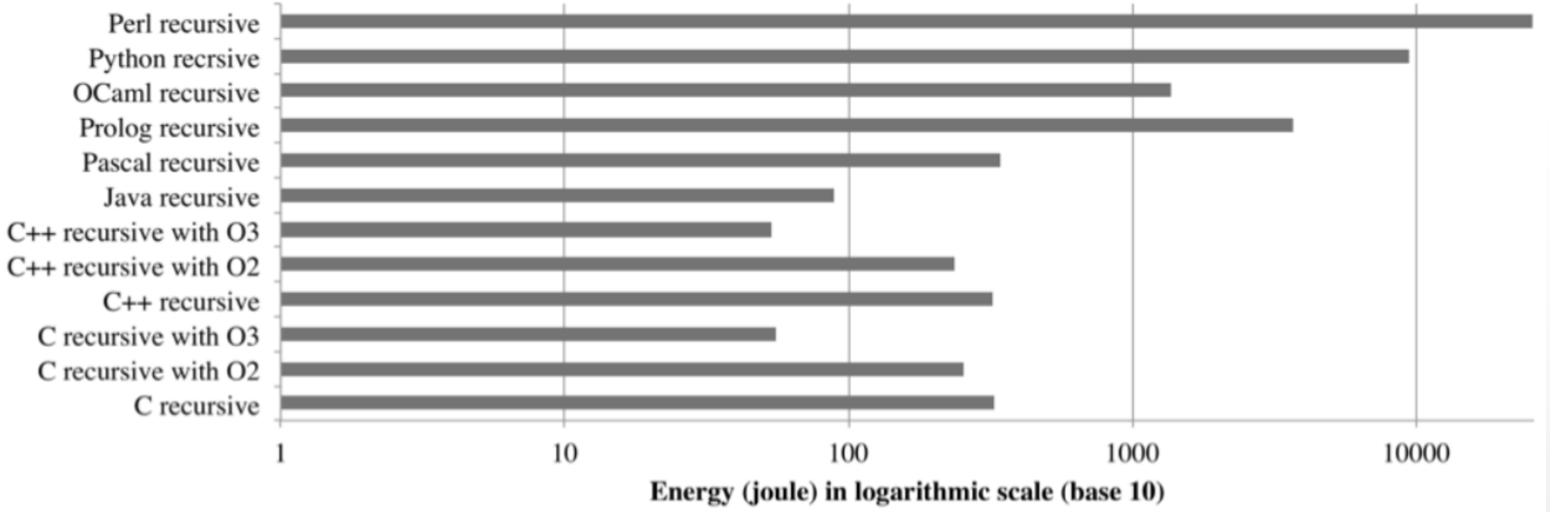
- **Code vectorization: Single Instruction Multiple Data (SIMD)**
  - **SIMD**: Perfom different operations with different data into the same instruction!
    - **Increase performance and reduce energy consumption!**



  - **Example for Intel:**
    - SSE (Streaming SIMD Externsions)
    - AVX (Advanced Vector Extensions) 128/256 bit instruction types

Programming language

- **Problem: Choosing an efficient programming language may lead to significant energy waste**
- **Programming languages** have a great impact on the performance and energy consumption!
  - Different causes: compiled or interpreted language, memory management, etc.
  - Depending on the level of abstraction of the language the user has more or less opportunities to introduce optimizations!
- Example with the Tower of Hanoi in different languages (Energy To Solution - ETS)



Choosing the most effient programming language is crucial energy efficient software!

Energy Efficient Libraries and Drivers

- **Problem: Not exploiting well-proven energy efficient solutions can lead to inefficient software!**

- Select the most efficient library routines to increase performance and reduce energy consumption!
  - Look at the use multithreading inside the routines
  - If necessary trace them from the performance and energy perspectives!

- Example with linear algebra routines to perform matrix decompositions:

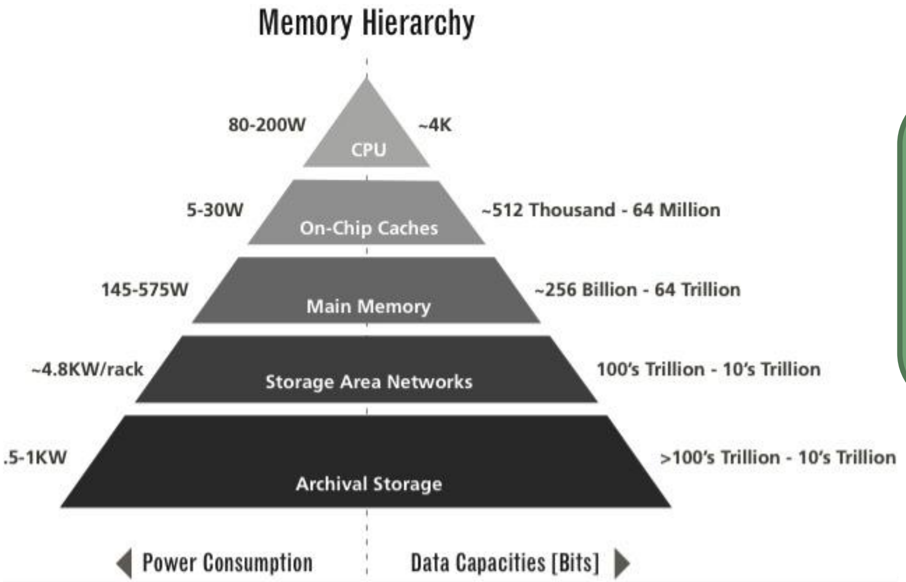
	LU factorization			Cholesky factorization			Reduction to tridiagonal form	
	LAPACK	MKL	SMPSs	LAPACK	MKL	SMPSs	LAPACK	MKL
T (s)	18.37	10.99	13.25	6.50	5.48	5.09	73.83	17.99
GFLOPS	38.96	65.13	54.02	55.06	65.31	70.31	1.24	5.09
P_{max} (W)	390.70	385.78	392.81	384.61	389.06	393.52	327.42	336.33
P_{min} (W)	301.64	294.37	328.12	307.27	289.92	292.04	285.00	297.89
P_{avg} (W)	359.72	377.94	385.56	373.13	377.80	373.73	293.87	325.95
P_{wrk} (W)	112.22	130.44	138.06	125.63	130.30	125.23	46.37	78.450
E_{tot} (J)	6,608.60	4,155.61	5,109.44	2,427.28	2,072.07	1,905.7	21,698.53	5,865.51
E_{wrk} (J)	2,061.48	1,433.54	1,829.30	816.60	714.04	643.65	3,423.50	1,411.32

TABLE I
PERFORMANCE, POWER AND ENERGY OF THE DIFFERENT IMPLEMENTATIONS FOR THE THREE DENSE LINEAR ALGEBRA OPERATIONS.

In the example it is always more efficient to use MKL Intel library!

Minimizing data movement

- **Problem: Unnecessary data movement may lead to energy wasting**
- **Energy efficient software should minimize data movement!**
  - Move data over short distances
  - Execute tasks with fewest memory accesses



An efficient memory hiererchy should store data as close as possible to the CPU!

- The less energy is consumed for a memory access, the closer dara is stored to CPU

Same energy: 1 access to RAM = 7 instructions executed in CPU = 40 cache accesses

- **A solution: buffer and bactch data requests in one operation/instruction**

Design Energy Efficient software!

- **Engineering practices...**
  - **Problem: Traditional software enguneering models do not support energy efficiency as a relevant concern**
    - **Solution**: The software life cycle should be optimized and energy efficiency be integrated as a non-funcional requirement into the software engineering process model

- **Energy Efficient software is still not well perceived!**
  - **Problem: Despite the fact that software can influence the energy consumption of HPC systems dramatically, the importance of software aspects of energy efficiency is still not perceived**
    - **Solution:** Encouraging further research and better education of all stakeholders of HPC systems

Evaluation of the seminar

- A topic for each student will be assigned
  - Individual presentations:
    - 30 slides (approx.)
    - 60 minutes + discussion
    - The slides should contain notes that clarify their content

- More information at:
  - http://wr.informatik.uni-hamburg.de/teaching/wintersemester_2014_2015/energy-efficient_programming
  - **Please register the mailing list:** http://wr.informatik.uni-hamburg.de/listinfo/eep-1415

- Contact and supervision:
  - **Dr. Manuel Dolz (manuel.dolz@informatik.uni-hamburg.de) - Coordinator**
  - Michael Kuhn (michael.kuhn@informatik.uni-hamburg.de)
  - Dr. Julian Kunkel (julian.kunkel@informatik.uni-hamburg.de)
  - Konstantinos Chasapis (konstantinos.chasapis@informatik.uni-hamburg.de)
  - *Prof. Dr. Thomas Ludwig (ludwig@dkrz.de)*

Thanks for you attention!

Questions?