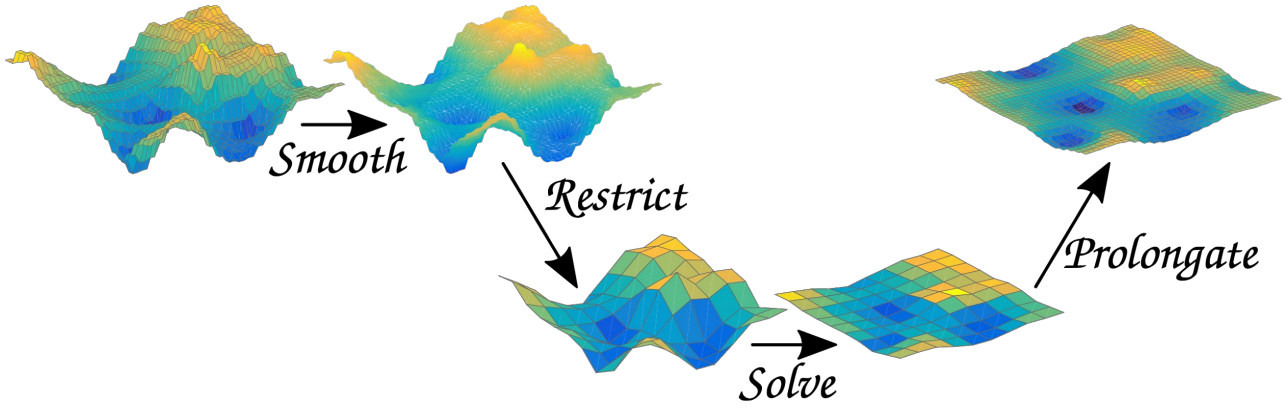
LyNcs: Linear Algebra, Krylov-subspace methods, and multi-grid solvers for the discovery of New Physics

Jacob Finkenrath
Simone Bacchio

CaSToRC, The Cyprus Institute



PRACE 6IP WP8 LyNcs: People

- Simone Bacchio, CaSToRC
- Jacob Finkenrath, CaSToRC
- Luc Giraud, Inria
- Michele Martone, LRZ
- Matthieu Simonin, Inria
- Shuhei Yamamoto, CaSToRC

- Constantia Alexandrou
- Kyriakos Hadjiyiannakou
- Giannis Koutsou
- Floriano Manigrasso

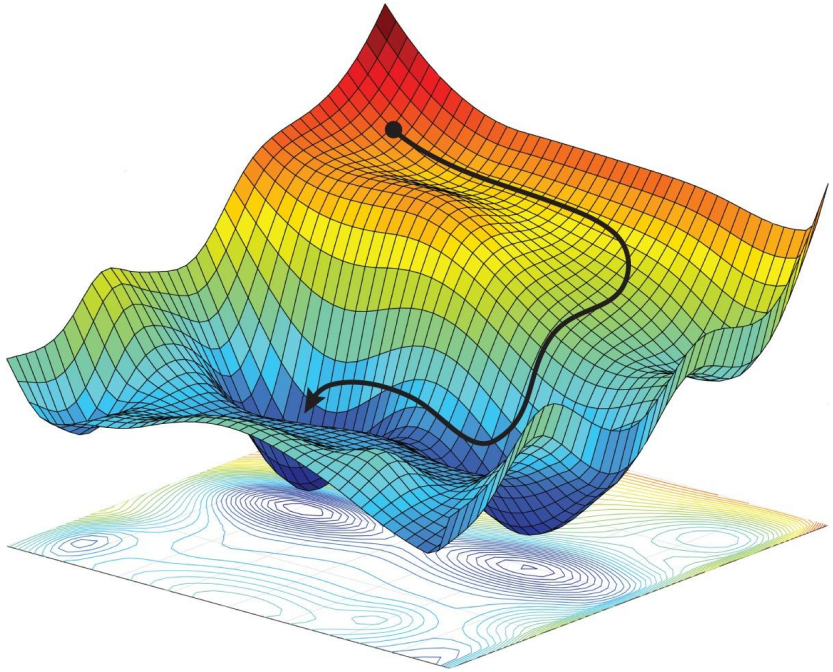
- Mike Chatzittofi
- Raphaella Demetriou
- Christodoulos Stylianou

THE CYPRUS INSTITUTE
RESEARCH·TECHNOLOGY·INNOVATION

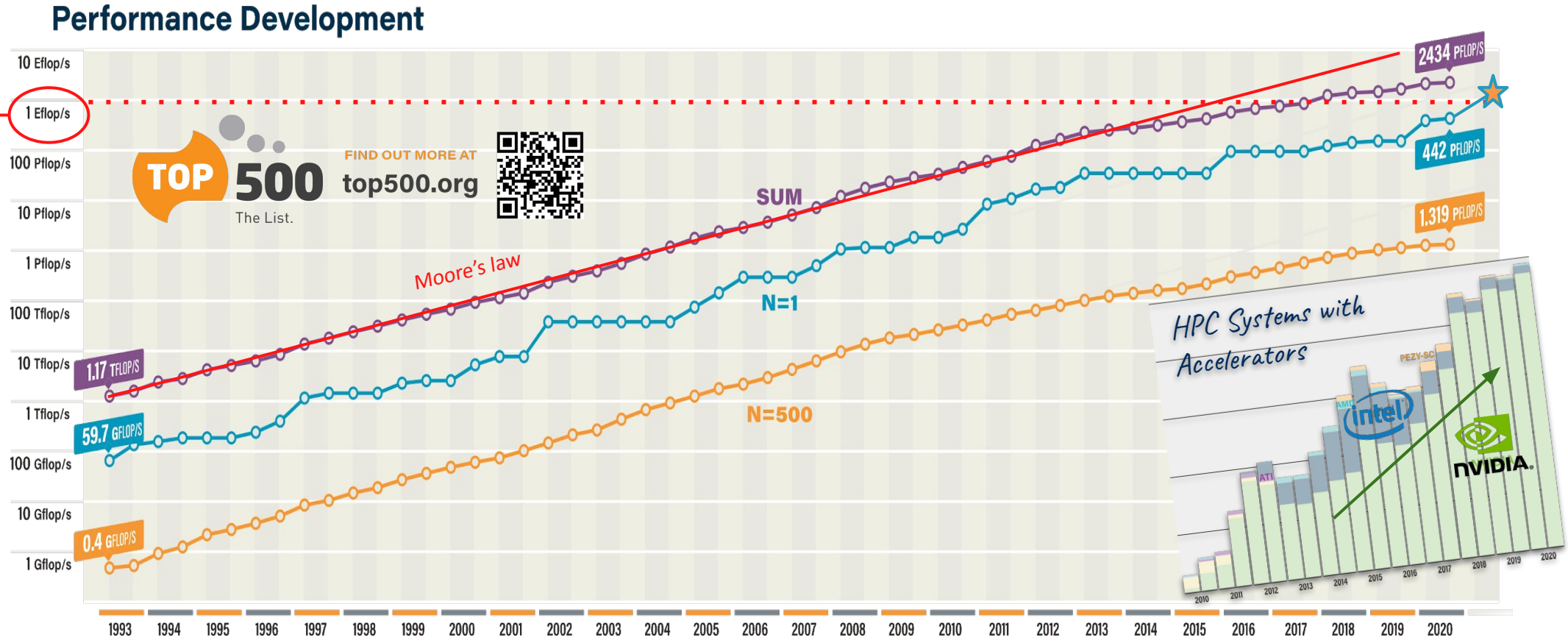
Leibniz Supercomputing Centre
of the Bavarian Academy of Sciences and Humanities

Talk Outline

- **Challenges in High Performance Computing**
- **Computation in lattice QCD**
  - Sparse Matrix vector multiplication
  - Multigrid methods
- **Solutions addressed by LyNcs**
  - Performance and Efficiency
  - Scalability
  - Diversity/heterogeneous HPC
- **Software Overview**

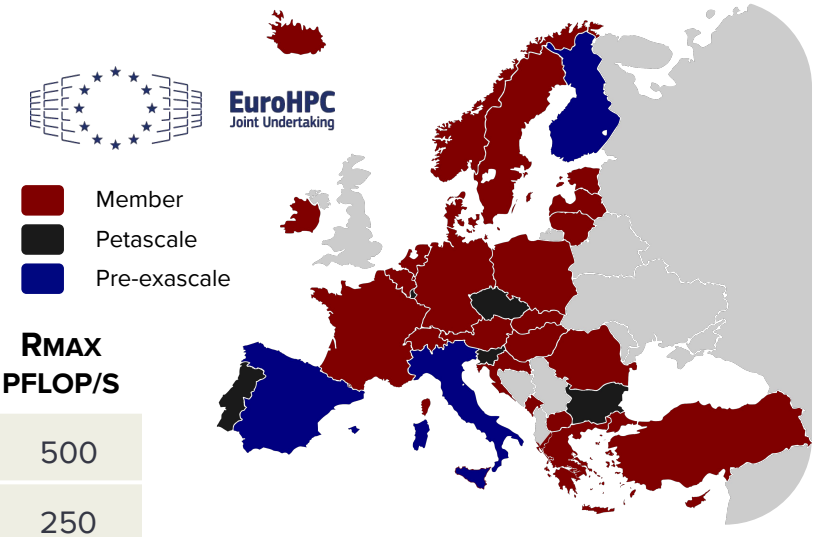


The next generation of supercomputers will break the barrier of exascale performance!



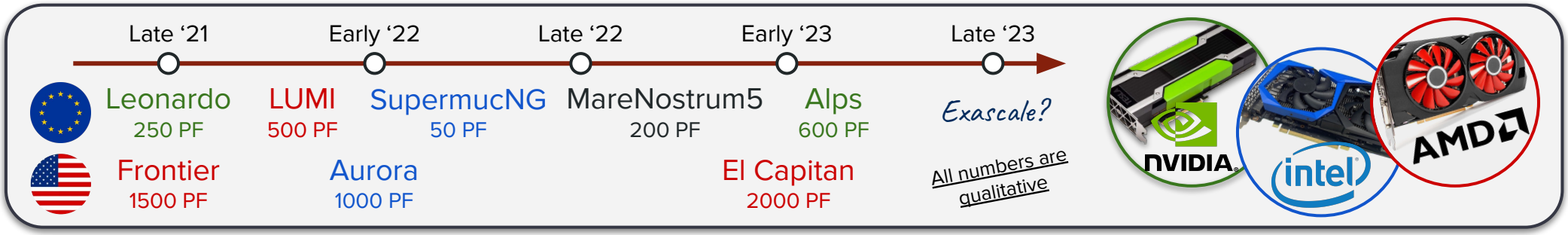
FLOP/s = Floating Point Operations per Second (in double precision)

Upcoming supercomputers



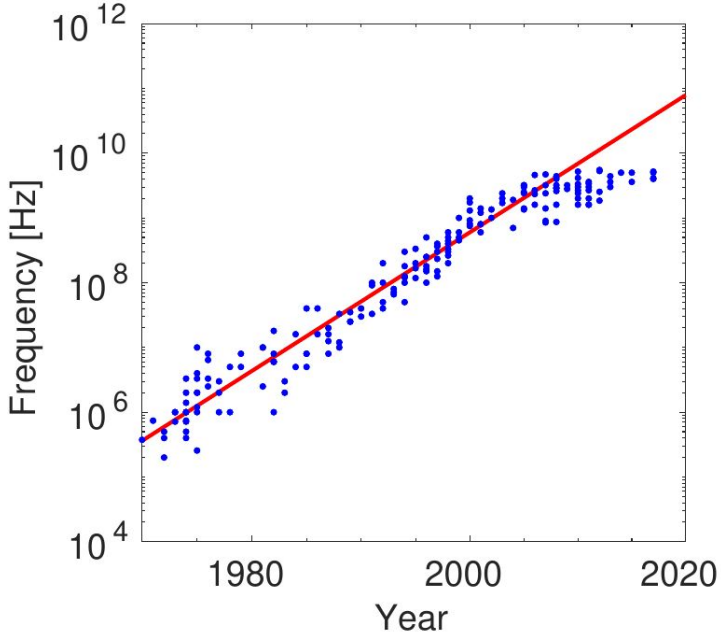
Pre-Exascale European systems

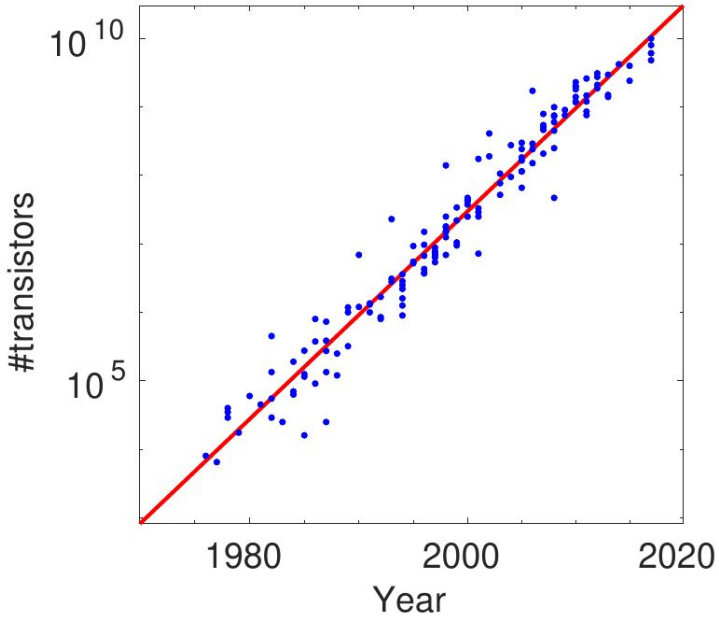
					RMAX PFLOP/S
2021	**LUMI**	AMD Epyc CPUs, AMD Instinct GPUs	CSC	Finland	500
2021	**Leonardo**	Intel Xeon CPUs, NVIDIA Ampere A100	Cineca	Italy	250
2022	**MareNostrum5**	Two clusters, heterogeneous system, TBA	BSC	Spain	200



- **Software suite need to be flexible to perform on different architecture**

Challenge: Post-Dennard and Moore scaling





Dennards Scaling is over:

- Clock Frequency is not increasing by 2x every 1.6 years

Not possible to simply wait that the application is becoming faster

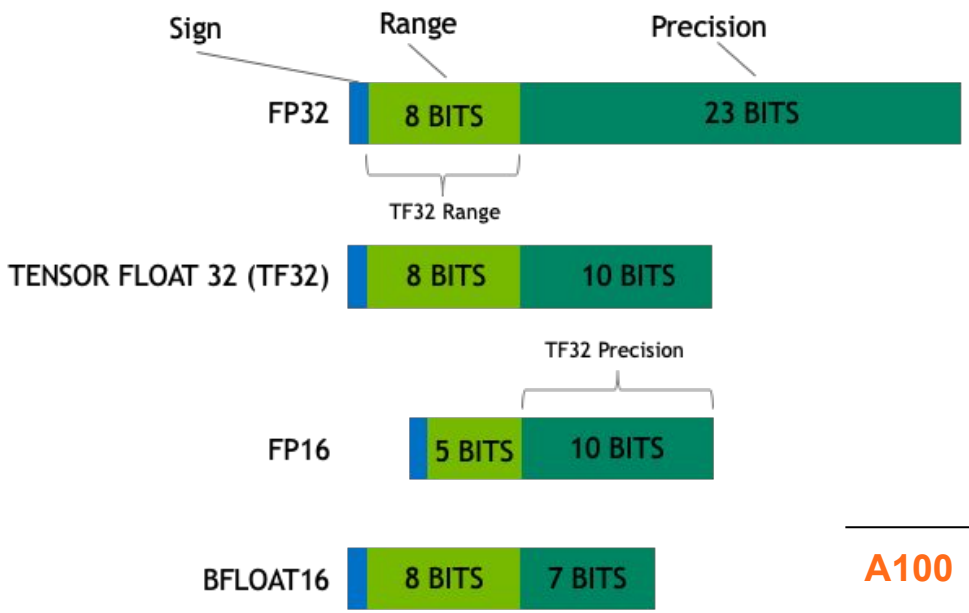
Moores Law is coming to an end

- Transistor density on chip is increasing by 2x every 2 year

Computing architectures are becoming more complex

- **Software needs to be massively parallelized**

Challenge: Complexity



	FP64	FP32	FP16	FP32 Tensor	BFLOAT 16	INT8	INT4
A100	9.7	19.5	312	156	312	624	1248
V100	7.8	15.7	125				
P100	4.7	9.3	18.7				

Trend of Nvidia GPUs:
towards low precision performance

useful for AI applications

- **Develop algorithms which can exploit low precision performance**

Acceleration of Research by High Performance Computing

An simple example:

- let's say we have an application which needs 24 hours to execute
- 10x speed-up by parallelization/optimization is possible

Can we afford to wait for 24 hours for every new job?

- imaging: other research groups have an optimized application
- they will be way ahead within the next year !!!

We would never catch up!

Is this an unrealistic example ?

It's exactly what is happening towards exascale computing:

16x

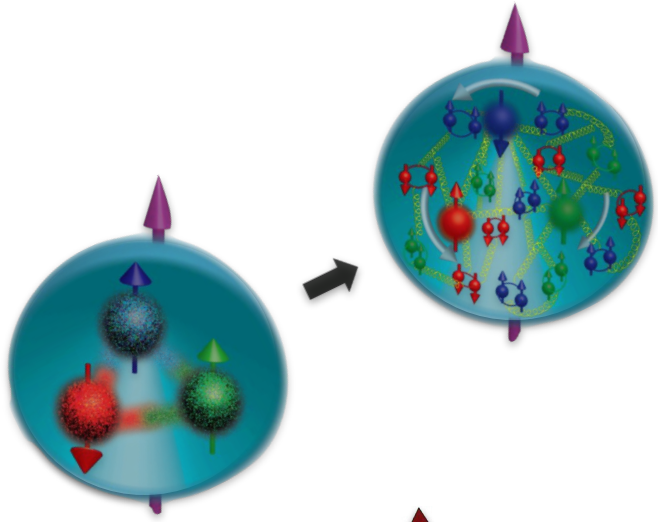
Fastest Machine in Europe:
JUWELS Booster ~ 70 PFlops

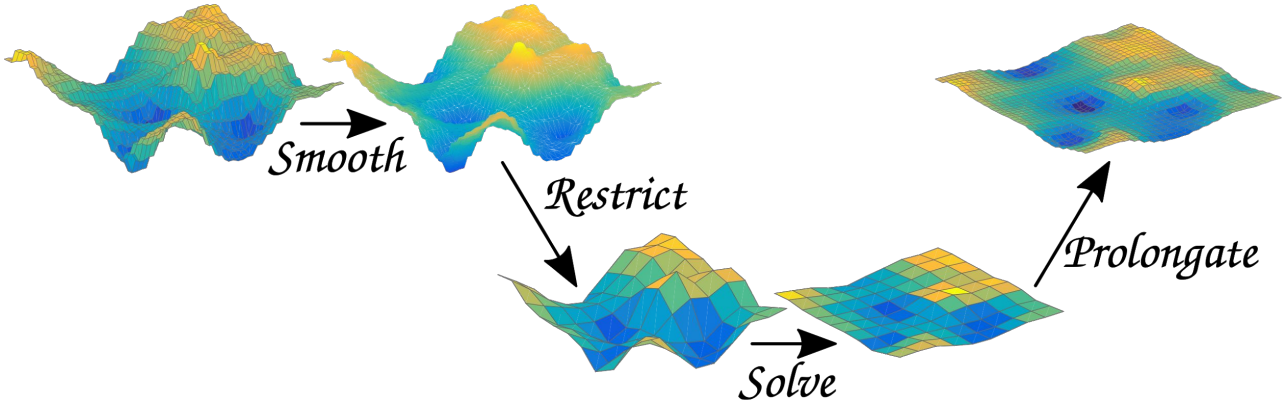
Fastest Machine in 2022:
Frontier > 1500 PFlops

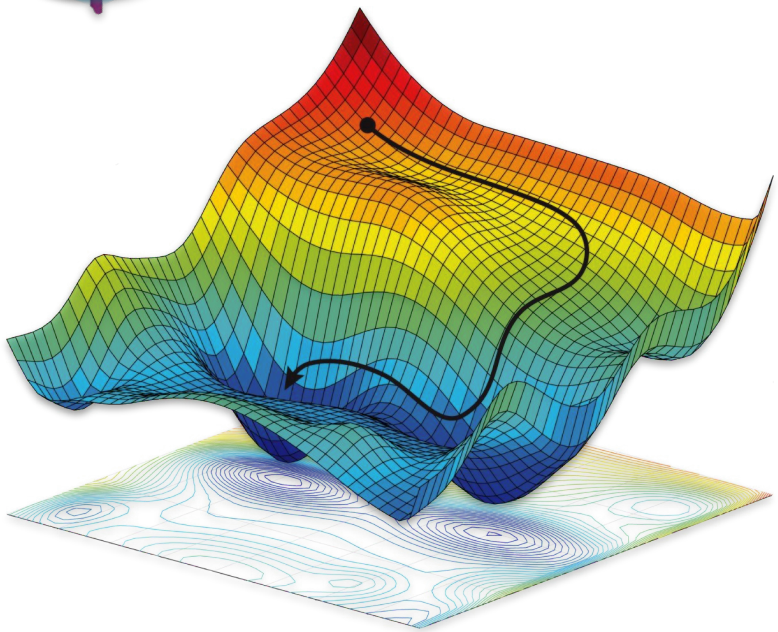
@Simone: What we need to address for lattice QCD to be ready ?

The Case of Lattice QCD

- Sparse Matrix vector multiplication
- Multigrid methods



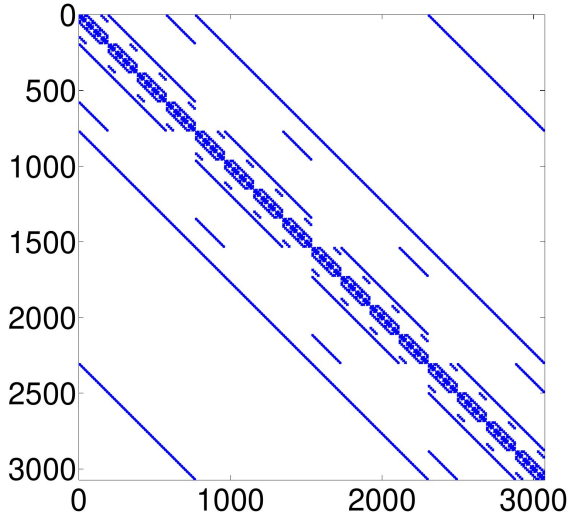




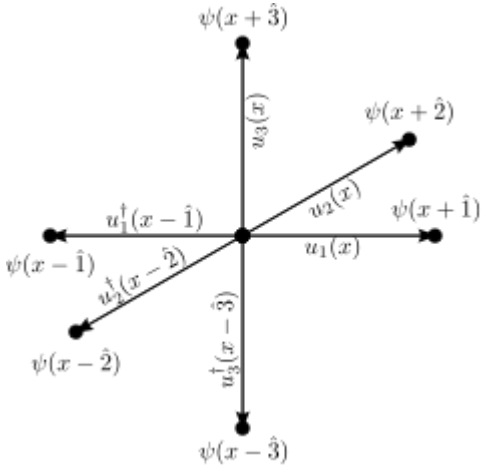
Linear equation

- Computational challenge solving:

$$Dx = b \qquad D \in \mathbb{C}^{n \times n} \text{ and } x, b \in \mathbb{C}^n$$



- 4-dim matrix stencil with *n ~ O(10^8) - O(10^9)*



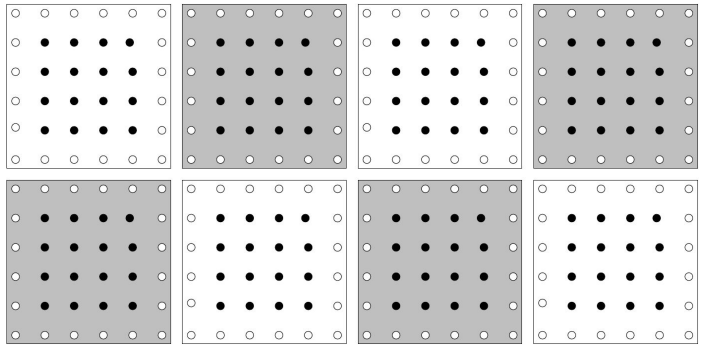
- Matrix vector stencil with nearest neighbor connections
- Iterative linear solver approach requires Matrix vector multiplications
- Matrix vector product bandwidth bounded:

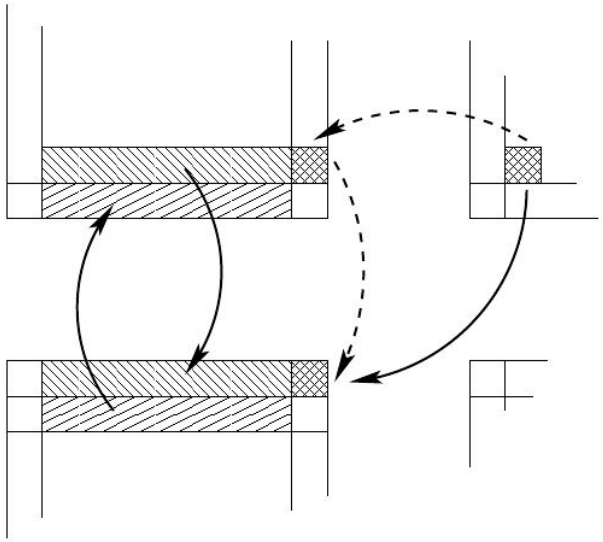
  Flops : Bytes = 1.3 : 1.1

Parallelization strategies for Matrix vector product

Domain Decomposition:

- split the lattice in subdomains
- for 16 = 4x4 processes follows:
  - local lattice on each proc is 16x smaller





Nearest neighbor interaction:

- creating memory buffers and communicate them before and after local multiplication

Computation:

1. send boundaries
2. inner multiplication
3. receive boundaries
4. outer multiplications

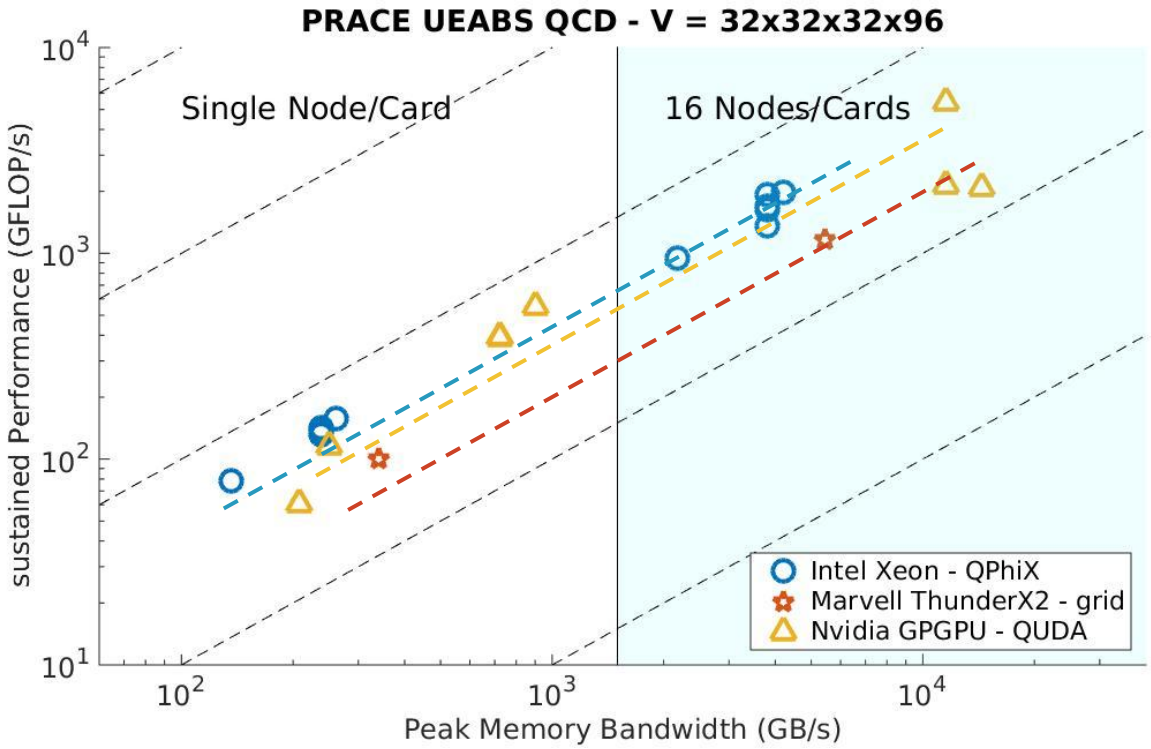
Performance of Matrix Vector product

UEABS Benchmark-kernels on
PRACE Tier 0 systems (Prace 4/5IP)

Application of lattice codes:

- QUDA for Nvidia GPUs
- QPHIX for Intel CPUs
- Grid for Arm

Computation cost dominated by
matrix-vector application



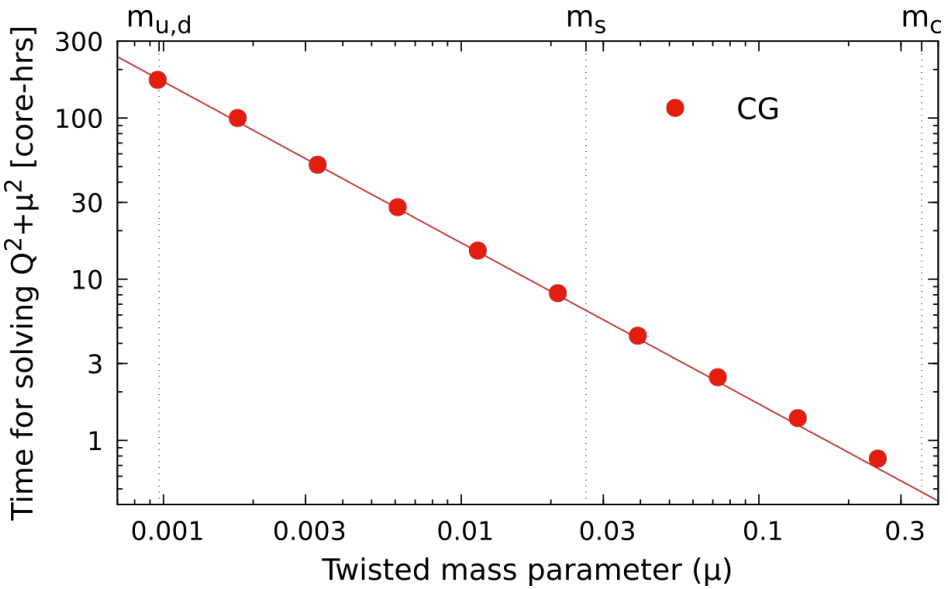
- computational performance is bounded by available bandwidth
- towards 16 nodes, Nvidia GPU deviates from perfect scaling

Upcoming: benchmarks on the current PRACE Tier 0 systems

Krylov Subspace Methods

- Computational challenge solving:

$$Dx = b \qquad D \in \mathbb{C}^{n \times n} \text{ and } x, b \in \mathbb{C}^n$$



- Iterative solver: Krylov methods

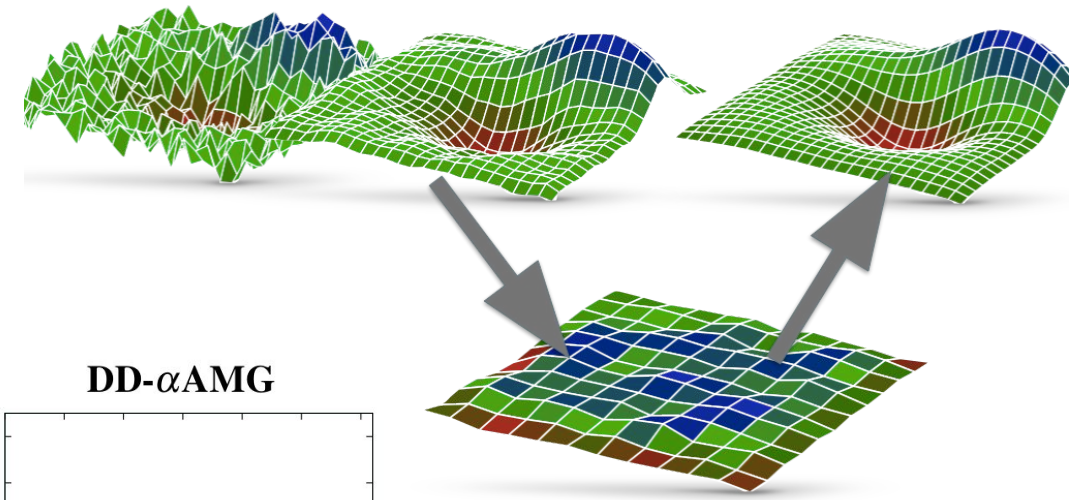
$$x \in \mathcal{K}_n(D,b) = \left\{b, Db, D^2b, \ldots, D^{n-1}b\right\}$$

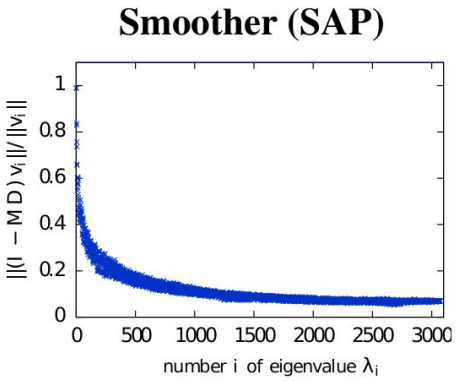
- Many variants and improvements available!
  - Standard iterative solvers: CG, GMRES, GCR, BiCGStab, …
  - Mixed precision solvers
  - Preconditioners
  - Block solvers
  - Communication-avoiding
  - Pipelined solvers
  - Deflated solvers
  - Multigrid methods

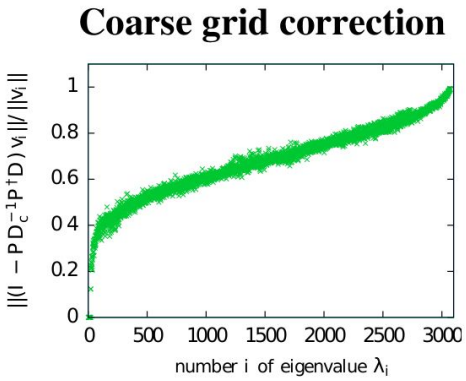
Idea: use multiple layers of coarsening for accelerating the inversions.

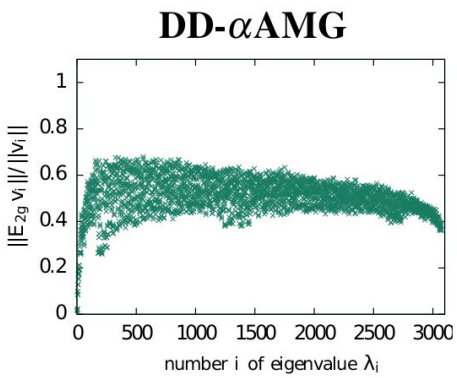
For the Dirac operator we have used Algebraic Multigrid: $D_c = RDP.$

Multigrid is used in combination with
a **smoother** as preconditioner
of a Iterative Krylov solver







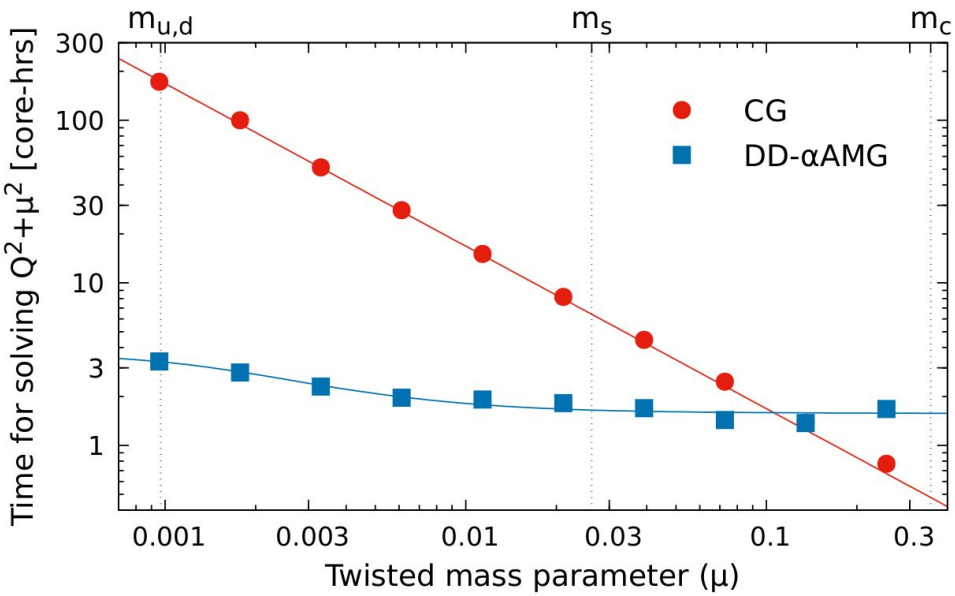


Multigrid solvers give a great speed-up compared to standard solvers:

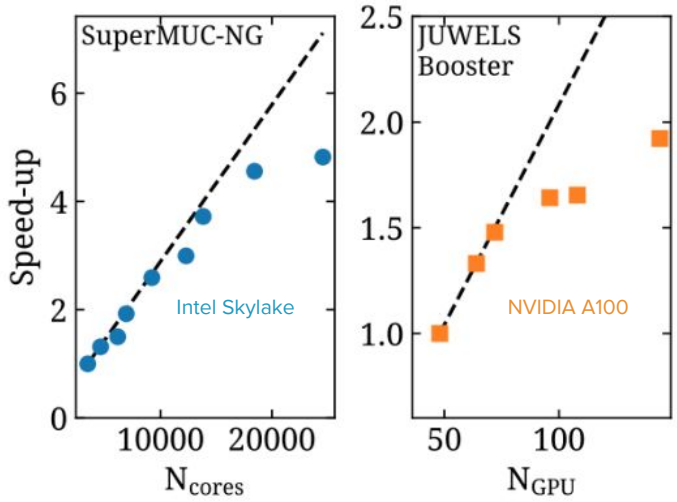
Up to 100x faster at the physical pion mass

<u>This problem has been very challenging:</u>

- Algorithmic-wise
- Tuning-wise (huge parameter space)
- Implementation-wise
- Performance-wise
  - Combination of computationally- and communication-intensive kernels
  - Difficult to scale on HPC systems, specially on GPUs



[C. Alexandrou, S. Bacchio, et al. Adaptive Aggregation-based Domain Decomposition Multigrid for Twisted Mass Fermions. Phys. Rev. D94(11):114509, 2016]



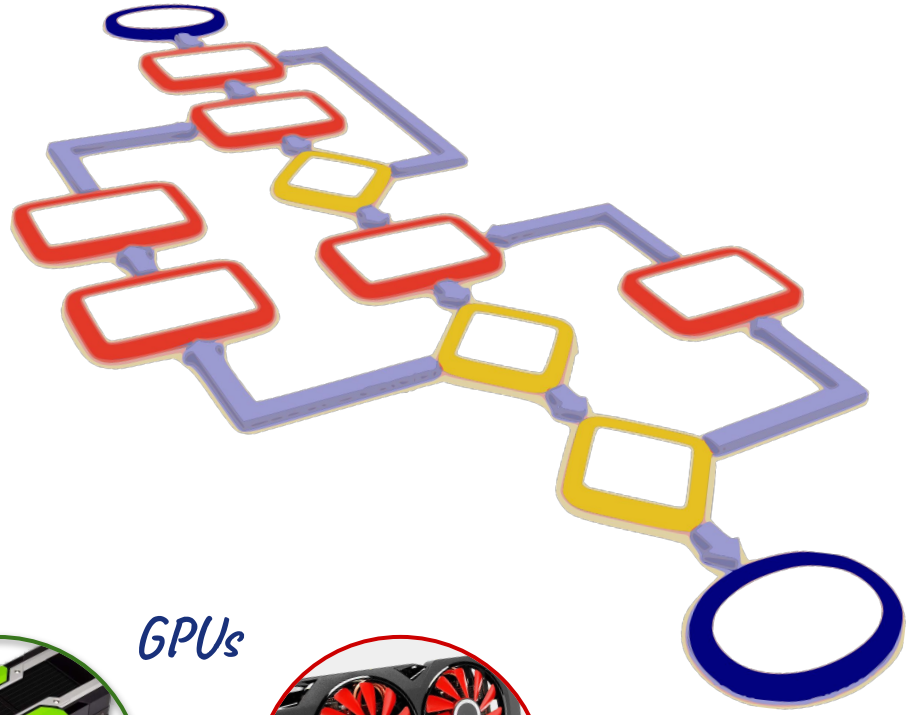
- **Target:** using 20% or more of an exascale system
  - Current status (based on Juwel Booster):
    - Single task scales up to 0.2% of such system
    - With parallel tasking we would use up to 4% of it
  - Going to exascale requires
    - Usage of communication-avoiding algorithms
    - Rethinking traditional parallelization strategies
    - Advanced task management and deployment
  - Our work in this direction is two-fold
    1. Research on algorithms suitable for exascale systems
    2. Development of software solutions for their ease use

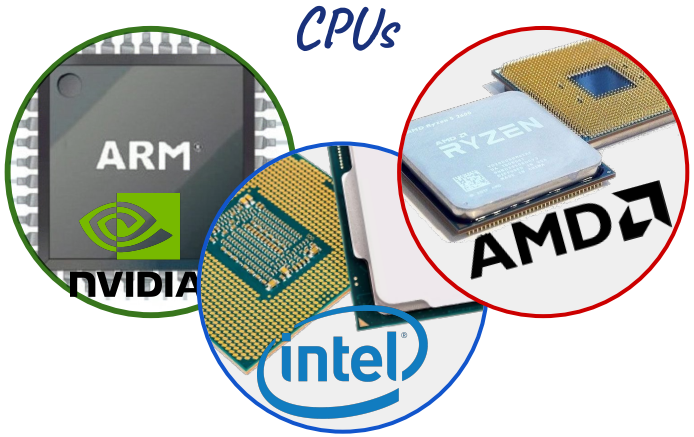
We are **top users** of the largest supercomputer in EU
Performance: 44 PFlop/s

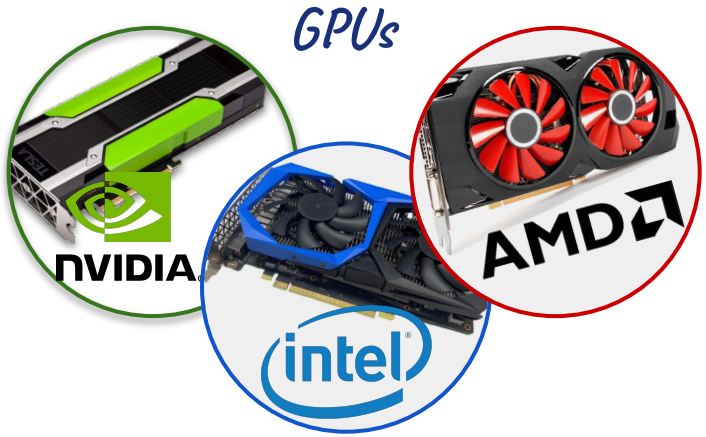
@Jacob: How did we plan to reach exascale performance?

The LyNCs project

- Performance and Efficiency
- Scalability
- Diversity/heterogeneous HPC







HPC software challenges are addressed within
PRACE 6IP WP8: Forward looking software solutions

- Deliver solutions in the form of high quality, transversal software that address challenges posed by the rapidly changing HPC pre-Exascale landscape to HPC users and scientific communities.
- Advance strategic projects, selected in a transparent and competitive process, that require a long-term and significant dedicated effort.
- Allow for disruptive approaches to modernize HPC software, leveraging of software technologies adopted, developed, and maintained by the big players in IT

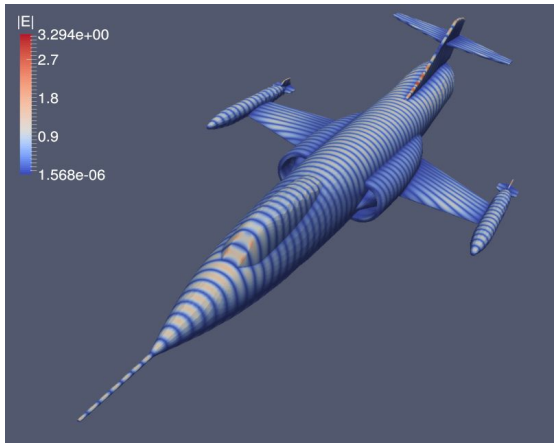
LyNcs is one of 10 selected projects out of 23 project

Project Objectives

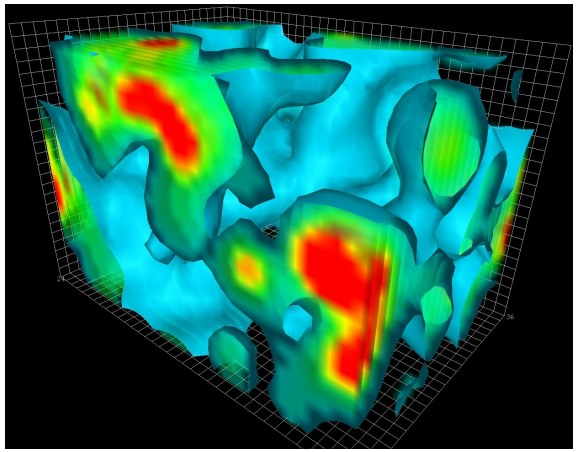
LyNcs: Linear Algebra, Krylov-subspace methods, and multigrid solvers for the discovery of New Physics

Address the Exascale challenge for sparse linear systems & iterative solvers

- Targeting all levels of software stack
  - on lowest level: Sparse BLAS kernels: *librsb*
  - on intermediate level: iterative linear solver libraries: *fabulous, DDalphaAMG, QUDA*
  - on highest level : community codes: *LyNcs API, tmLQCD*



http://www-sop.Inria.fr/nachos/index.php/Software/HORSE



http://www.physics.adelaide.edu.au/theory/staff/leinweber/VisualQCD/ImprovedOperators/

Applications in:

- Fundamental Physics - lattice QCD
- Electrodynamics
- Computational Chemistry

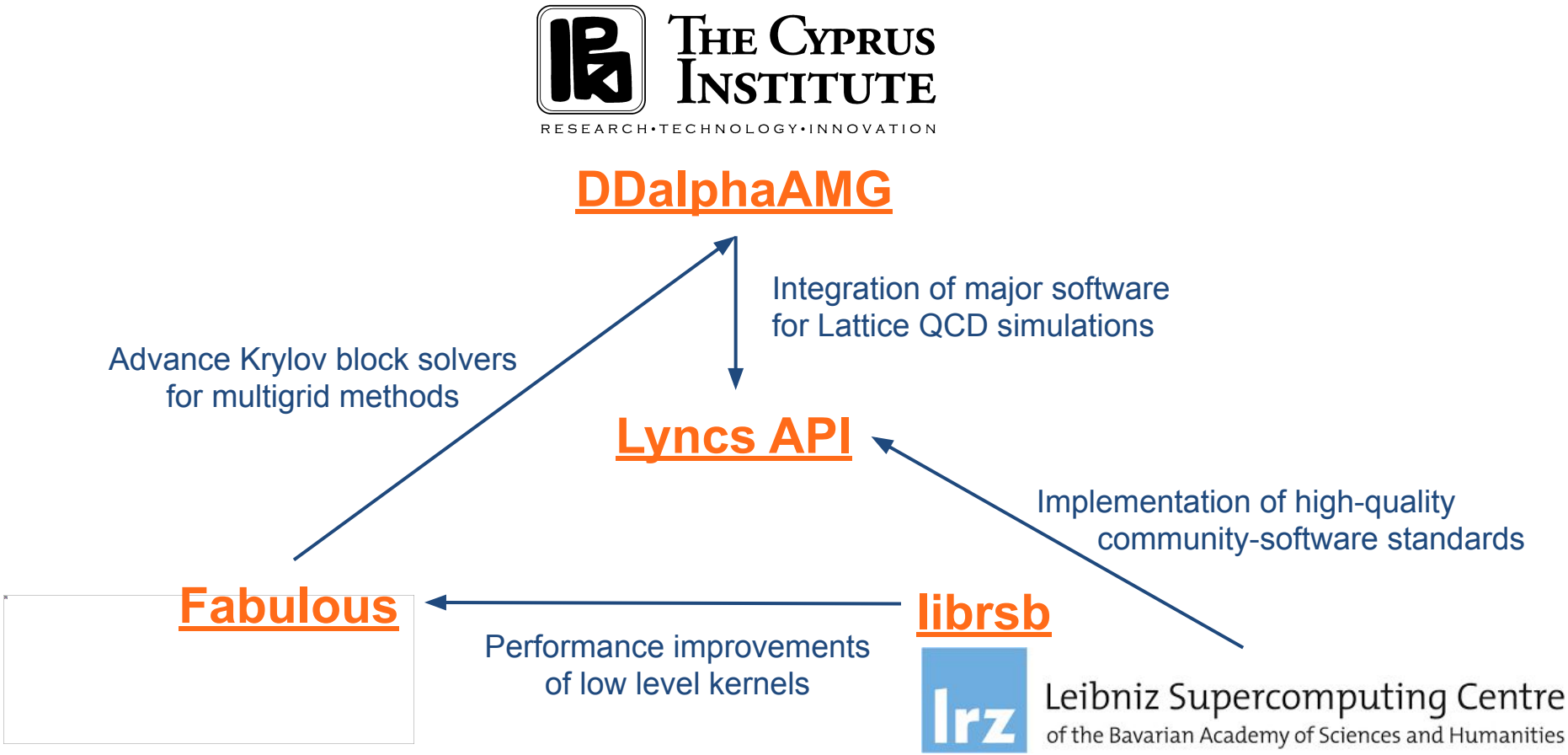
LyNcs vision: Towards Exascale Computing

- **Objective 1: Performance and Efficiency**
  - focusing on efficient node level performance by using state of the art implementations and algorithms like Block Krylov solvers and multigrid approaches
- **Objective 2: Scaling**
  - extend scalability and speeding up the time to solution beyond Domain Decomposition, namely strong scaling with alternative parallelization strategies
- **Objective 3: Diversity**
  - Hardware and software: Software stack of community software is rich and optimized for various hardware systems, optimizations and algorithm developments are on going in various tasks like given by DOEs Exascale initiative

Focus of LyNcs is to be complementary/in a synergy to those efforts

- on user-friendly API to enable the rich software stack of solver and application codes
- on specific software optimization and developments used in the European Community

Project Synergy



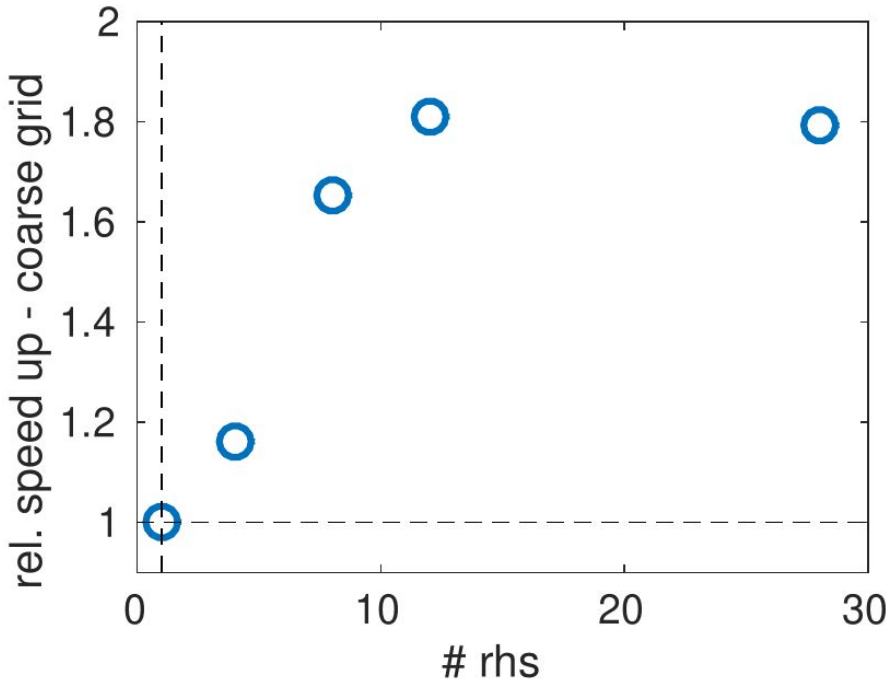
Algorithmic improvements and code refactoring

DDalphaAMG multiple RHS - multigrid library of LQCD

- Increase Efficiency : Decrease required bandwidth to computing ratio
- Enhance Portability : Utilization of portable compiler optimization rather than architecture-specific
- Improve Convergence : Using Block-Krylov solvers from Fabulous

Implementation:

- Definition of a new data structure, index on vectors runs the fastest
  - re-writing of all low level kernels
- using auto-vectorization to enhance portability
  - `# pragma unroll`
    `# pragma vector aligned`
    `# pragma ivdep`
  - on Skylake chips this shifts vectorization from 128 bit to 256 bit
  - coarse grid computation speed up by a factor ~1.8

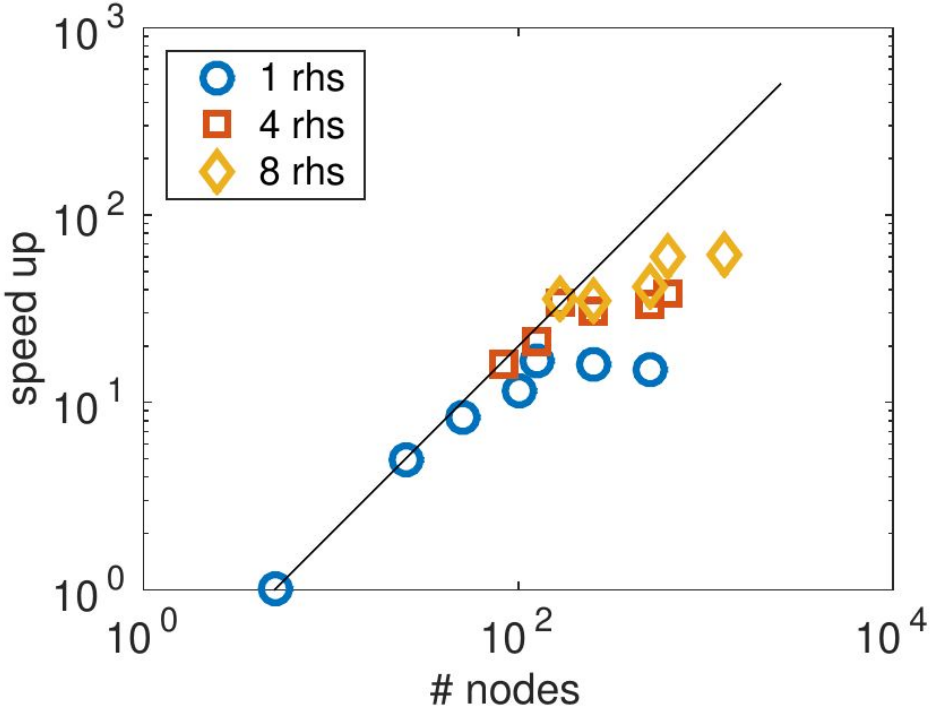


Performance and Scalability

Multiple righthand sides are suitable:

- **Extend of Strong scaling window**
  - strong scaling test on SuperMUC-NG using a V=160x80x80x80 lattice
  - extend strong scaling regime by around x5

- **Preparation for pipeline force computation in Hybrid Monte Carlo**
  - multiple right hand sides with multiple shifts
  - force computation of different operators can be done together
    - better usage of bandwidth and vectorization, increase of efficiency, utilization of mix-precision



Contribution to Objectives

- O2: Algorithm development to extend strong scaling regime
- O3: Optimization of computing kernel to add flexibility for target architecture

Fabulous - Block Krylov solver library

Block Minimum Norm Residual Techniques

- Solve: $A \cdot X = B$

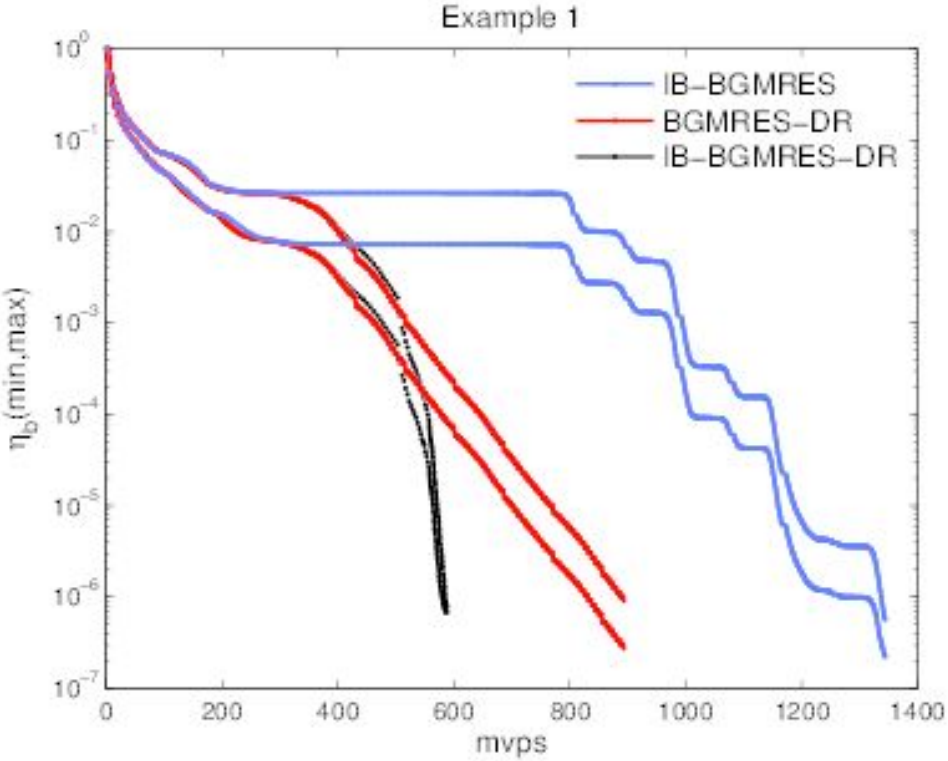
  for $A \in \mathbb{C}^{n \times n},\ X, B \in \mathbb{C}^{n \times p}$ and $p \ll n$

- Search space:
  - Look for approx. in nested space V_k

$$X_k = \text{argmin}||B - AX_k||_F$$

Main Difficulties

- Individual convergence rate for each rhs
  - rank deficiency in search space (inexact breakdown)
- Storage of the search space basis
  - restart by keeping infos from search space (spectral information recycling)



Contribution to Objectives

- O1: Algorithm development to utilize node level performance
- O2: Algorithm development to extend strong scaling regime

Performance and Scalability

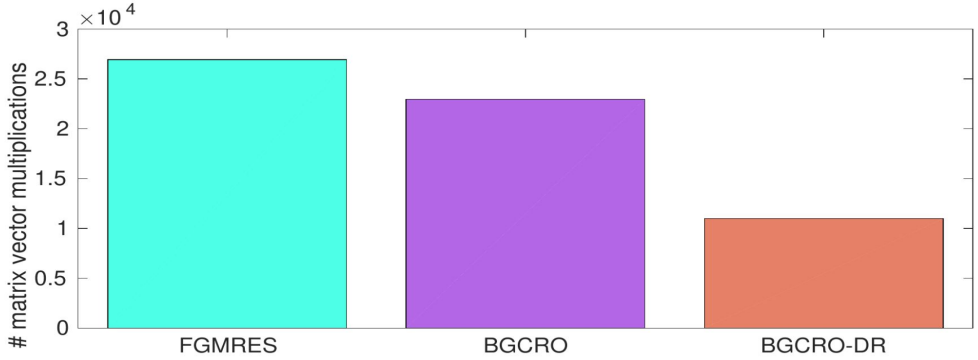
- **DDalphaAMG + fabulous**
  - utilization of Block Krylov solver on coarsest level

Usage of BGCRO-DR at L=32 lattice

- 2x at coarse grid shifts of cmu = 5
- 10x without coarse grid shifts

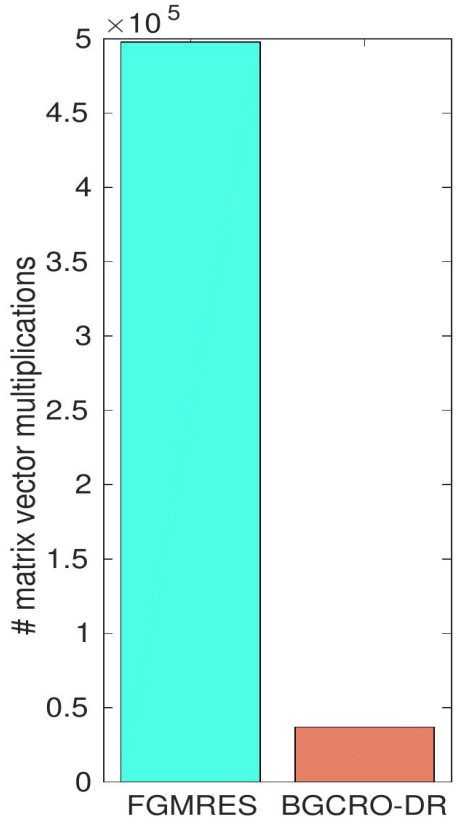
Fine tuning of large parameter space is ongoing

(currently Fabulous has some overhead)



BGCRO-DR is

effective without coarse grid shits

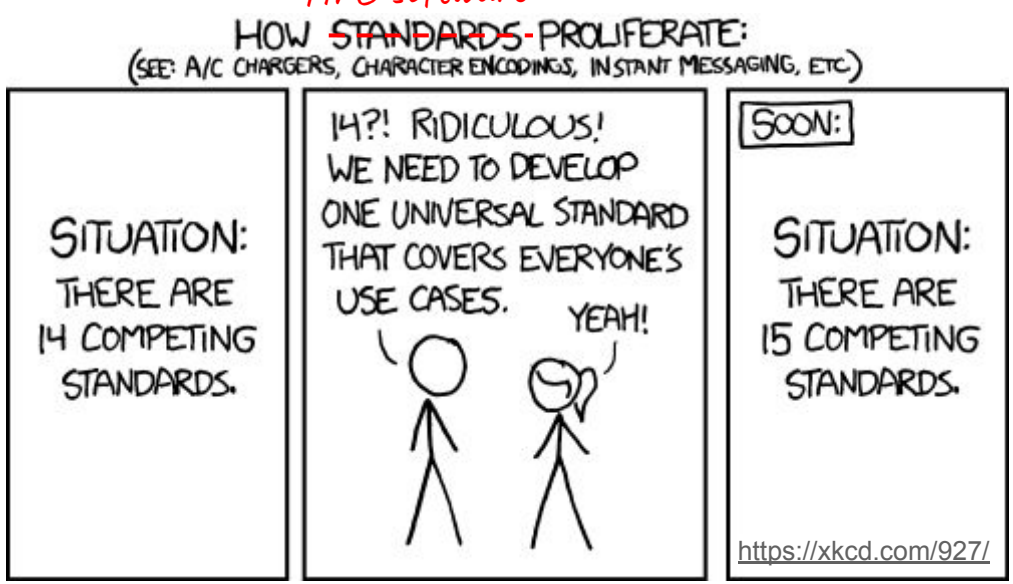


@Simone: Is there an easy way to utilize this algorithmic developments in an application code ?

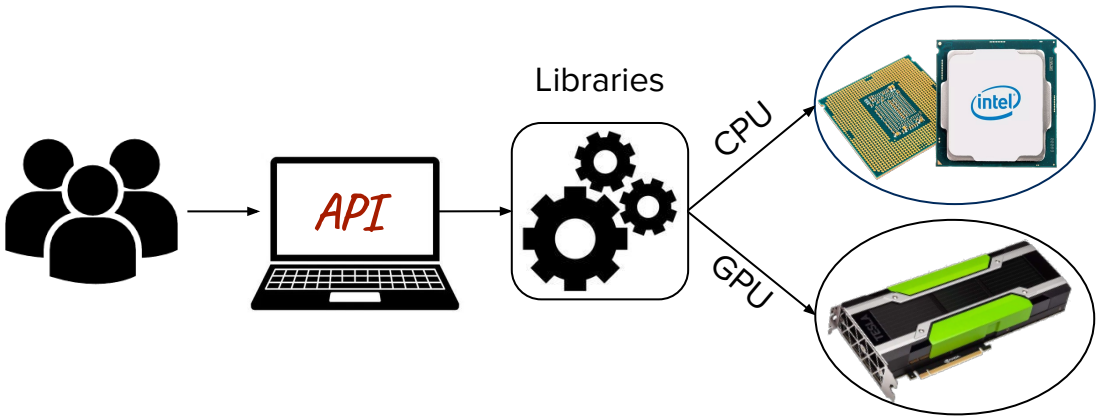
Software developments

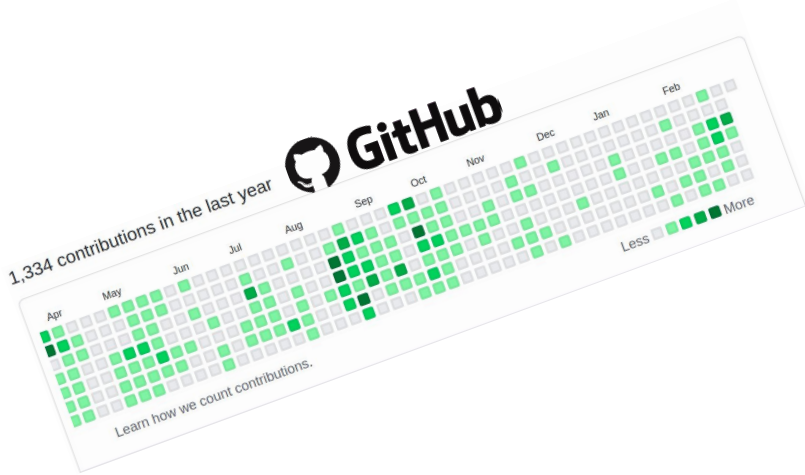
- Performance portability
- Flexibility & modularity
- Python & user-friendliness

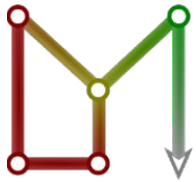
HPC software



Think twice before starting something new!





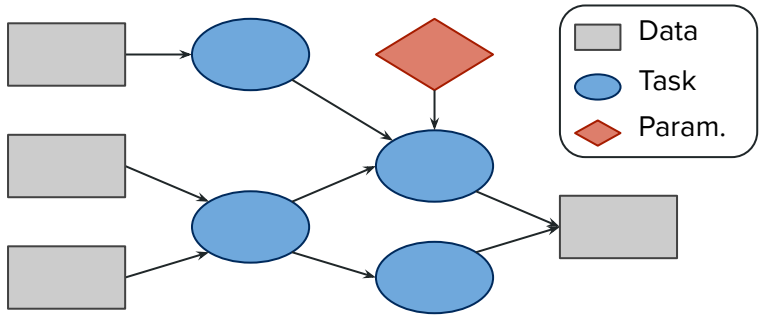


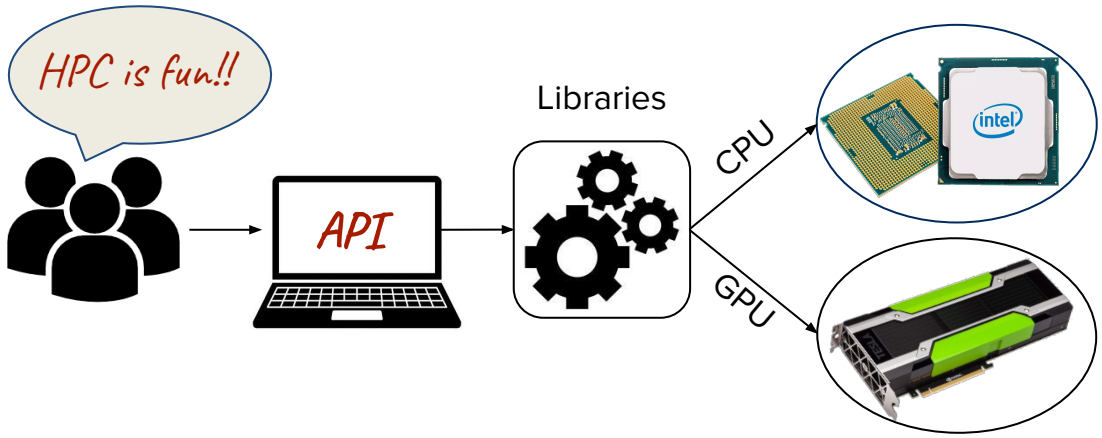
Lyncs-API

A Python API for Lattice QCD applications

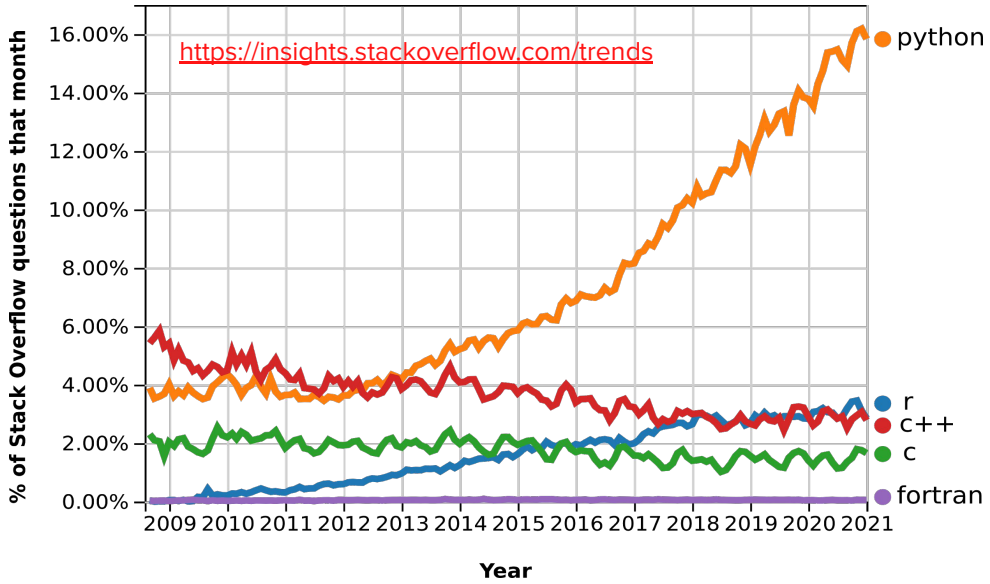
https://lyncs-api.github.io/ s.bacchio@gmail.com

- Usage of optimized libraries
  - Focus on CPU and GPU implementations
  - With architecture-dedicated optimizations
- Parallel tasking and modular computing
  - Usage of both CPUs and GPUs
  - Advanced task management
- Advanced algorithms for linear systems
  - Communication-avoiding and pipelining
  - Local and asynchronous approaches
- High-level software solutions
  - Python for HPC and APIs
  - Portability and user-friendliness





- Python is the most used language in data science
- Success thanks to
  - Being user-friendly and interpreted
  - Dedicated software solution
    - Numpy, scipy, and pandas
    - Tensorflow, keras and pyTorch
- Usage in HPC
  - Very active research field
    - Solutions for parallel data and tasking
    - Optimization of performance
    - Automatic binding to C/C++ libraries
    - Seamlessly compilation of code
- Easy integration of machine learning models!



Flexible, portable, extendible

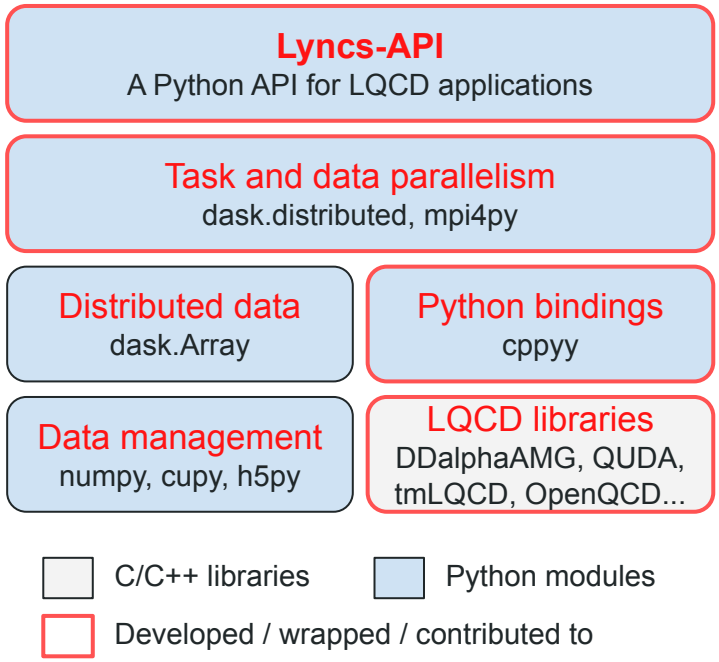
➢ Lyncs-API covers the entire stack, from low-level libraries to fully-fledged community codes

Main features:

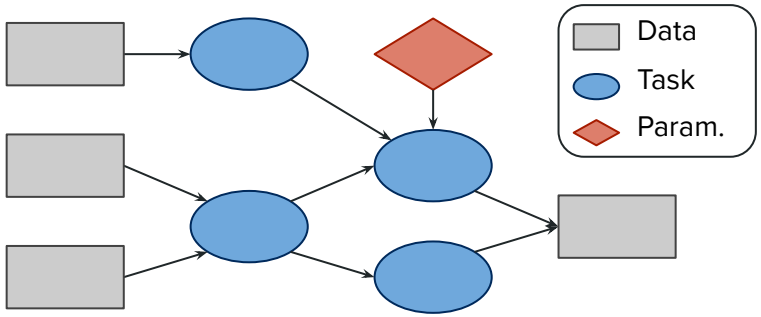
- On github: https://github.com/Lyncs-API
- **Modularized:** single interfaces distributed as independent packages
- High-standards enforced (CI/CD, linting, docs, etc...)
- Python integrated and compatible

Main target:

➔ Task-distributed simulations on CPUs & GPUs for ETMC



1. **Lyncs Interface**
2. **Computational Graph**



3. **Scheduler + Tuner**
   a. **Single-node** (threads, processes, memory)
   b. **Modular** (CPU, GPU, etc.)
   c. **Distributed** (remote task & data management)

DASK

Natively scales Python

- Management of distributed data and tasks
- Task scheduling and parallelism via futures
- Distributed numpy arrays

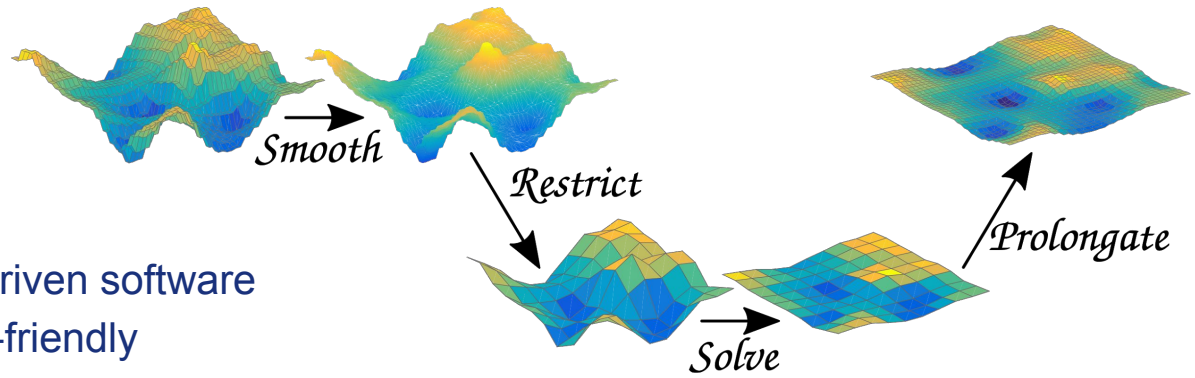
cppyy

Automatic C/C++ bindings

```
>>> cppyy.include('header.h')
>>> cppyy.load_library('lib.so')
```

- Direct access to library functions
- Automatic casting of Python types
- Support of templates and overloading

Summary of Target Applications:



Our software implementation in LyNcs:

Target 1: **LyNcs API** - New community-driven software
➔ portable, flexible, modular and user-friendly

new: python API enable modular and parallel task computing

Target 2: **Fabulous** - Iterative Block Krylov Solvers library
➔ study and implementation of new algorithms

new: Block Krylov variants, spectral recycling

Target 3: **librsb** - Recursive Sparse Blocks format library
➔ improved portability, benchmarking and code quality

new: coverage, performance enhancement and multiple rhs

Target 4: **DDalphaAMG** - Multigrid library for lattice QCD
➔ extension to block solver and linkage to Fabulous

new: enable multiple right hand side implementation and various CPU optimizations

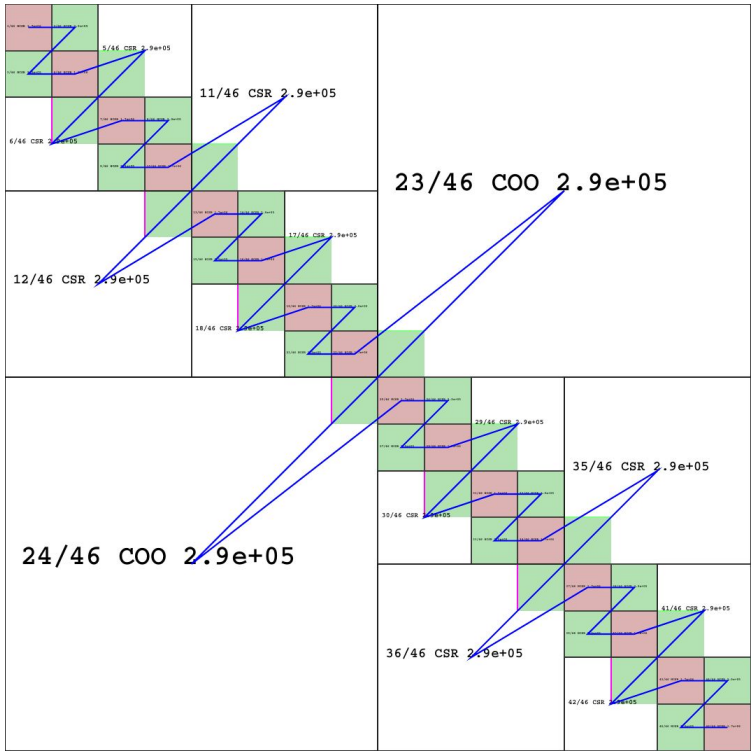
Leibniz Supercomputing Centre
of the Bavarian Academy of Sciences and Humanities

The Cyprus Institute
RESEARCH•TECHNOLOGY•INNOVATION

LIBRSB Heterogeneous Computing

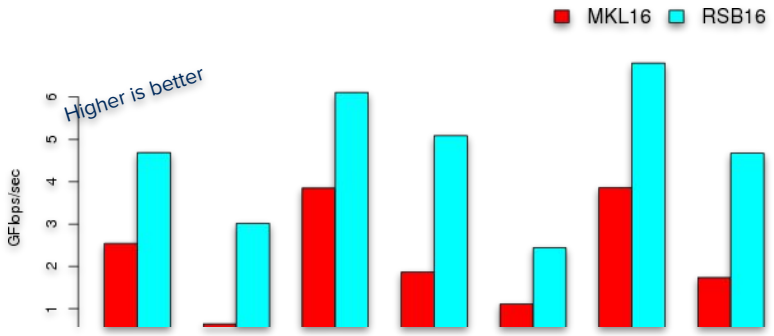
New software developments are done using CI/CD

- establishing Infrastructure :
  - datasets, prototype systems for testing our code, via LRZ to Beast
- adds flexibility to our developments in order to adapt to new hardware trends



Librsb : open source Sparse BLAS library available under http://librsb.sourceforge.net/

- Portable library
  - C/C++
  - Fortran
  - Python
  - Matlab/Octave
  - Julia
- Optimized performance



Lattice QCD on LUMI (AMD GPUs)

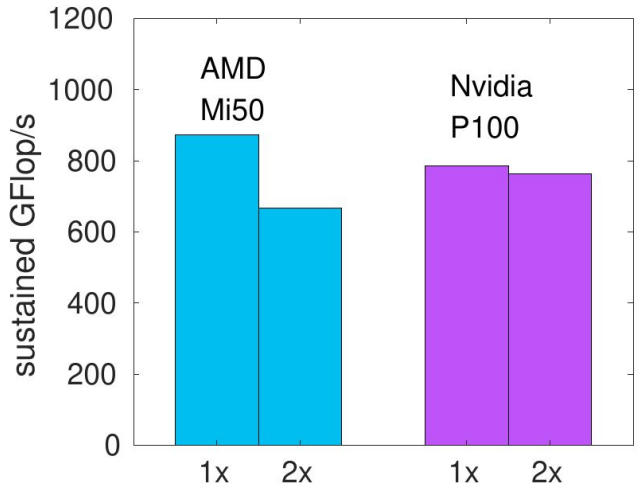
Achieve Readiness for the next Top European System

Towards LUMI ready software at the end of the year:

- using HIP port of QUDA develop under DoE exascale project and add specific capabilities
- currently tests are ongoing on Goethe-HLR at CSC at Uni of Frankfurt with AMD Mi50

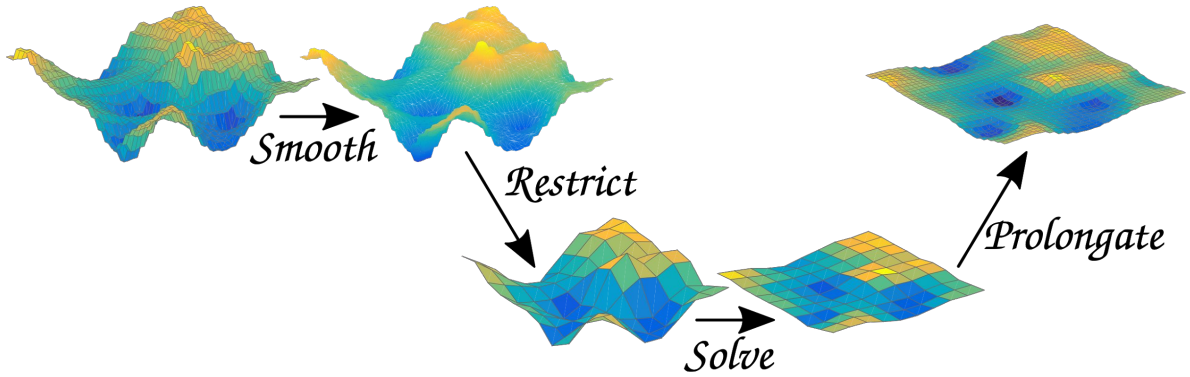
Ongoing and next steps:

- Performance evaluation of kernels
- Access to prototype system with AMD Mi100 at CSC Finland
- HIP port of application software



First Results from Quda Dslash benchmark kernel compare to Nvidia architecture

LyNcs: Conclusion



LyNcs is addressing exascale challenges on all levels:

- on the different software hierarchy : low level to highest level
- on the different exascale objectives: Scalability, flexibility, efficiency

and is on the way towards running on the European Pre-exascale machines

For application of Multigrid solvers in other areas, like in renewable energies:

- Talk by Dr. Pasqua D'Ambra at 13. July on:
  Algebraic MultiGrid Preconditioners for Sparse Linear Solvers at Extreme Scales on Hybrid Architectures

THANK YOU FOR YOUR ATTENTION

www.prace-ri.eu