Quantum Chronodynamics Simulations and Efficient Evaluation of the Wilson-Dirac Operator

$$D\psi(x) = A\psi(x) - \frac{1}{2}\sum_{\mu=0}^{4}\{[(I_4 - \gamma_\mu) \otimes U_{x,\mu}]\psi(x+\hat{\mu}) + [(I_4 + \gamma_\mu) \otimes U^{\dagger}_{x-\hat{\mu},\mu}]\psi(x-\hat{\mu})\}.$$

Claude Tadonki

Mines ParisTech
Centre de **R**echerche **I**nformatique
claude.tadonki@mines-paristech.fr

- **Q**uantum **C**hromo**D**ynamics (**QCD**) is the theory of strong interactions, whose ambition is to explain nuclei cohesion as well as neutron and proton structure, i.e. most of the visible matter in the Universe.

  ➡ Creation of the universe (matter synthesis right after the BIG BANG) !!!

- The main computational model for Quantum chromodynamics (QCD) investigations is the so-called **L**attice **Q**uantum **C**hromo**D**ynamics (**LQCD**).

  ➡ Based on the LQCD model, (large scale) simulations are implemented !!!

- In the LQCD model, the space-time is represented by a finite discrete system with **cartesian coordinates**, while the interaction between subparticles is governed by strong force theory.

  ➡ LQCD provides an analytical formalism, the reference for numerical studies.

- With a **fixed lattice spacing**, **fine-grained** simulations are considered through **larger lattice** volume, thus increasing **memory** and **computation** complexities.

  ➡ Solving the Wilson-Dirac system is the most critical computation kernel.

- The Wilson-Dirac operator D on a site x (a spinor) can be defined as

$$D\psi(x) = A\psi(x) - \frac{1}{2}\sum_{\mu=0}^{4}\{[(I_4 - \gamma_\mu) \otimes U_{x,\mu}]\psi(x+\hat{\mu}) + [(I_4 + \gamma_\mu) \otimes U^{\dagger}_{x-\hat{\mu},\mu}]\psi(x-\hat{\mu})\}$$

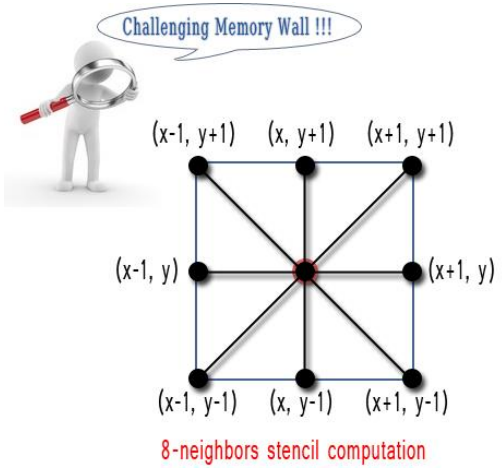
where

- A is a 12×12 complex matrix of the form $\alpha I_{12} + \beta(\nu \otimes \gamma_5)$, where α, β are complex coefficients and ν a 3×3 complex matrix
- x is a given point of the lattice, which is a finite subset of $\mathbb{N}^4$
- ψ (called *quark field* or *Wilson vector*) is a vector of 12-components complex vectors over the whole lattice
- $U_{x,\mu}$ is a 3×3 complex matrix (called *gluon field matrix* or *gauge matrix*) at (x, μ).

$$\gamma_0 = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix} \quad \gamma_1 = \begin{pmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & -i & 0 \\ 0 & i & 0 & 0 \\ i & 0 & 0 & 0 \end{pmatrix}$$

$$\gamma_2 = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix} \quad \gamma_3 = \begin{pmatrix} 0 & 0 & -i & 0 \\ 0 & 0 & 0 & i \\ i & 0 & 0 & 0 \\ 0 & i & 0 & 0 \end{pmatrix}$$

$$\gamma_5 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$



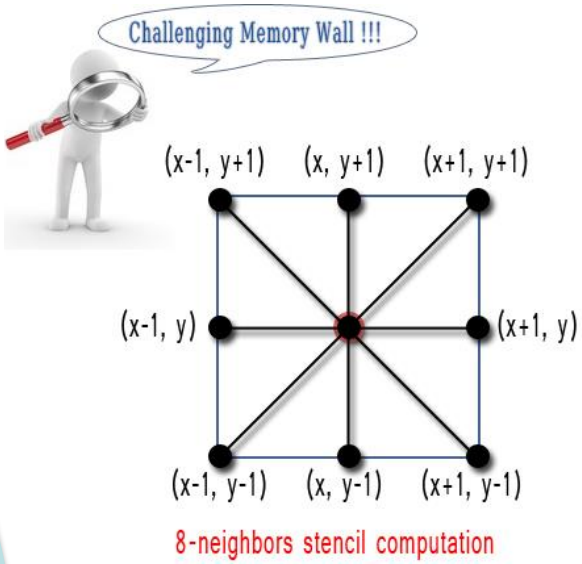
- The application of the Wilson-Dirac operator over all *spinors* of a *quark field* can be viewed as the product of a matrix-vector product, thus the ***Wilson-Dirac matrix***.

Remark 1. Given u a 4-components complex vector, $s \in \{-1, 1\}$, and $\mu \in \{0, 1, 2, 3\}$, the 4-components complex vector v defined by

$$v = (I_4 - s\gamma_\mu)u,$$

can be more efficiently calculated using the following relations

	$\mu = 0$	$\mu = 1$	$\mu = 2$	$\mu = 3$
$s = 1$	$v_1 = u_1 + u_3$ $v_2 = u_2 + u_4$ $v_3 = v_1$ $v_4 = v_2$	$v_1 = u_1 + iu_4$ $v_2 = u_2 + iu_3$ $v_3 = -iv_2$ $v_4 = -iv_1$	$v_1 = u_1 + u_4$ $v_2 = u_2 - u_3$ $v_3 = -v_2$ $v_4 = v_1$	$v_1 = u_1 + iu_3$ $v_2 = u_2 - iu_4$ $v_3 = -iv_1$ $v_4 = iv_2$
$s = -1$	$v_1 = u_1 - u_3$ $v_2 = u_2 - u_4$ $v_3 = -v_1$ $v_4 = -v_2$	$v_1 = u_1 - iu_4$ $v_2 = u_2 - iu_3$ $v_3 = iv_2$ $v_4 = iv_1$	$v_1 = u_1 - u_4$ $v_2 = u_2 + u_3$ $v_3 = v_2$ $v_4 = -v_1$	$v_1 = u_1 - iu_3$ $v_2 = u_2 + iu_4$ $v_3 = iv_1$ $v_4 = -iv_2$



Remark 2. Using the *normal factors decomposition* and its commutativity, we have

$$\begin{aligned}(I_4 - s\gamma_\mu) \otimes U &= ((I_4 - s\gamma_\mu) \otimes I_3)(I_4 \otimes U) \\ &= (I_4 \otimes U)((I_4 - s\gamma_\mu) \otimes I_3)\end{aligned}$$

Thus, the computation of $[(I_4 - s\gamma_\mu) \otimes U]\psi$ can be done as follows

$$[(I_4 - s\gamma_\mu) \otimes U]\psi = (I_4 \otimes U)\{((I_4 - s\gamma_\mu) \otimes I_3)\psi\}$$

- Using the above factorizations & simplifications leads to an optimal evaluation of Wilson-Dirac.
- Data locality is the main issue that will impact on memory and data communication costs.

8×N spinors	$8 \times (24 \times 8) \times N$	1536×N bytes
8×N U matrices	$8 \times (18 \times 8) \times N$	1152×N bytes
8×N integer indexes	$8 \times (18) \times N$	64×N bytes

Fig. Data inventory for a single Dirac operator

N = LxLyLzLt

- In order reduce the cost of irregular memory accesses related to SU(3) matrices, the so-called GAUGE-COPY strategy is considered: for each site of the whole lattice, the required 8 SU(3) matrices are stored contigously → better memory performance at the expense of redundancy!

- Examples of GAUGE array sizes (*sizeof(U) = 3x3x2x8 = 144 bytes*).

L	T	Sizeof(Us)
16	32	0.28 GB
32	64	4.5 GB
64	128	72 GB
128	256	1152 GB

- Parallelism is considered also because of the global memory requirement , even if work inefficient!!!

- Efficient Wilson-Dirac is vital for LQCD simulations as it is evaluated so many times.

Our goal is to solve the following linear system

$$D\psi(x) = \phi$$

- This system is an important step the synthesis of a statistical gauge configuration sample.
- The solution of the system, called propagator, appears in the expression of the so-called ***fermionic force***, used to update the momenta associated with the gauge fields along a trajectory in the **H**ybrid **M**onte **C**arlo (**HMC**) algorithm.
- Many propagators are computed on a trajectory, they should fullfil the **reversibility criterion**.
- Large-scale simulations involve huge Dirac matrices with bad condition numbers for small pion masses.
- Because of the reversibility condition and the quality expected for the propagators in order to be useful for physic, a high numerical precision is required to solve the Dirac system.
- Because of the implicit form of the Dirac matrix and the required precision, **iterative methods** are considered. We should prepare for a huge number of iteration or divergence.
- Large-scale inversions need a combination of state-of-the-art HPC and matrix computation !!!
- LQCD research is actiove through collaborations (projects) and dedicated supercomputers.

- Large volume of data (disk / memory / network)
- Significant number of solvers iterations due to numerical intractability
- Redundant memory accesses coming from interleaving data dependencies
- Use of double precision because of accuracy need (hardware penalty)
- Misaligned data (inherent to specific data structures)
  - Exacerbates cache misses (depending on cache size)
  - Becomes a serious problem when consider accelarators
  - Leads to « false sharing » with Shared-Memory paradigm (Posix, OpenMP)
  - Padding is one solution but would dramatically increase memory requirement
- Memory/Computation compromise in data organization (e.g. gauge replication)

- **High-precision L**attice **Q**uantum **C**hromo**D**ynamics simulations.
- The ANR project **PetaQCD** was targeting **256×128³** lattices.
- One evaluation of the *Dirac operator* on a **256×128³** lattice involves

$256 \times 128^3 \times 1500 \approx 10^{12}$ (stencil) **floating-point operations**

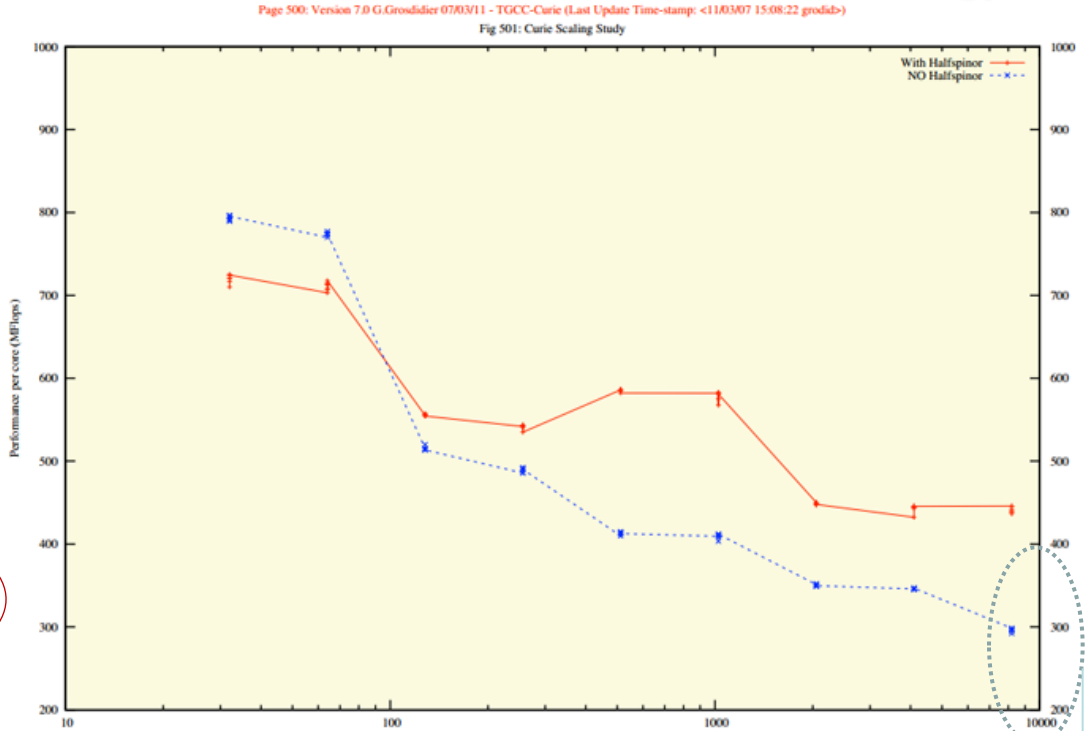
With our 10,000 cores, we can roughly perform $500 \times 10^3 \times 10^6 = 5 \times 10^9$ fps

Our 256×128^3 lattice would then require 200 seconds ≈ **3 minutes** for each evaluation of the *Dirac operator*.

10 days !!!

Now, imagine that we have to do it 5000 times to solve one Dirac linear system !!!

Curie Fat performance (weak scaling)

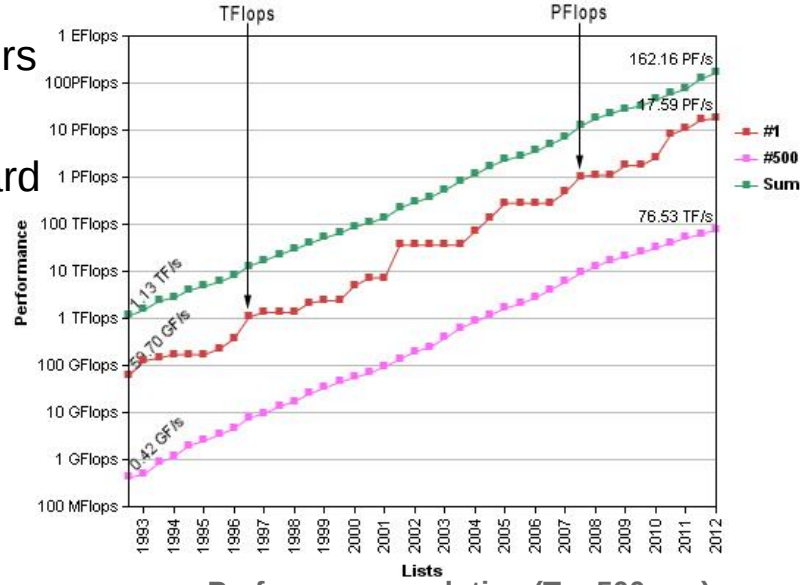


G.Grosdidier, « *Scaling stories* », PetaQCD Final Review Meeting, Orsay, Sept. 27th – 28th 2012

500 Mflops/core

- The (peak/sustained) performance of supercomputers is increasing significantly (≈ x10^5 since 1993).

  The following characteristics are becoming a standard
  - Several cores (2012 list: 84% ≥ 6 cores and 46% ≥ 8 cores)
  - Vector units (with larger vector registers)
  - Accelerated (mainly GPU: 62 systems of the 2012 and #1)



Performance evolution (Top500.org)

- The **gap between peak and sustained** performances on real-life applications is clearly questionable.

  ☞ HPC investigations should focus on this.

- When it comes to accelerators, **data transfer** is critical.

  ☞ Compromise and overlap (scheduling)

Chris Gregg and Kim Hazelwood, *Where is the Data? Why You Cannot Debate CPU vs. GPU Performance Without the Answer*, International Symposium on Performance Analysis of Systems and Software (ISPASS), Austin, TX. April 2011.

- Taking advantage of all aspects of a computing nodes requires a complex **hybrid/heterogeneous programming** .

  ☞ Code generation and transformation

- **Energy** is a major concern.

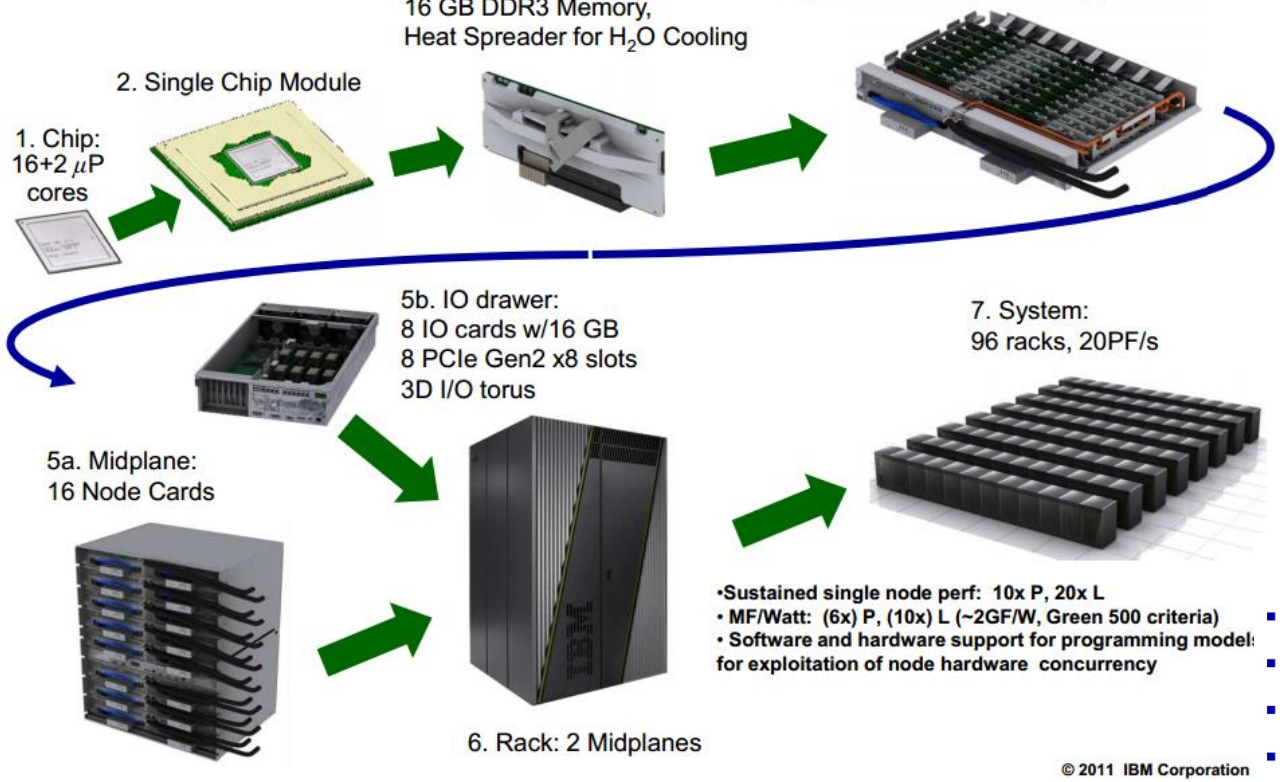
  ☞ Power aware programming and scheduling.

TITAN CRAY-XK7 the (2012) world fastest supercomputer
- 299 008 CPU cores (16-cores AMD Opteron 6274)
- 18 688 NVIDIA Tesla K20 GPUs
- Peak: **27.11** PFlop/s.
- Sustained: **17.59** PFlop/s (Linpack)

IBM System Technology Group

Blue Gene/Q

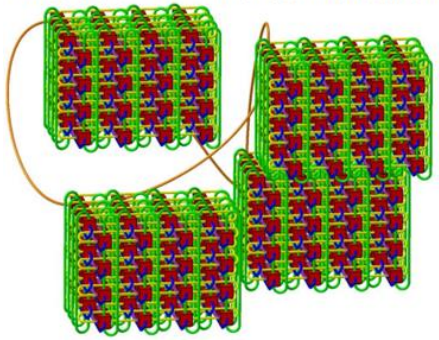
1. Chip: 16+2 μP cores
2. Single Chip Module
3. Compute card: One chip module, 16 GB DDR3 Memory, Heat Spreader for H_2O Cooling
4. Node Card: 32 Compute Cards, Optical Modules, Link Chips; 5D Torus
5a. Midplane: 16 Node Cards
5b. IO drawer: 8 IO cards w/16 GB 8 PCIe Gen2 x8 slots 3D I/O torus
6. Rack: 2 Midplanes
7. System: 96 racks, 20PF/s



- Sustained single node perf: 10x P, 20x L
- MF/Watt: (6x) P, (10x) L (~2GF/W, Green 500 criteria)
- Software and hardware support for programming models for exploitation of node hardware concurrency

NETWORK CHARACTERISTICS

- Integrated 5D torus at 1.76 GB/s (bidirectional)
- 2 GB/s raw bandwidth on all 10 links
- Hardware latency 80ns to 3us
- Additional 11th link for communication to IO nodes



Network Performance

- All-to-all: 97% of peak
- Bisection: > 93% of peak
- Nearest-neighbor: 98% of peak
- Collective: FP reductions at 94.6% of peak

- Nov 2011 TOP500 Entry

- # of cores: 65,536
- # of nodes: 4096 (4 racks)
- R_{max}: 677 TF
- R_{peak}: 838.9 TF
- N_{max}: 2719743 — very important for intensive simulations
- Power: 85 kW (network excluded)
- Sustained perf: 80.72% — Good, but this is linpack LQCD is another story !!!
- GF/W: 1.99

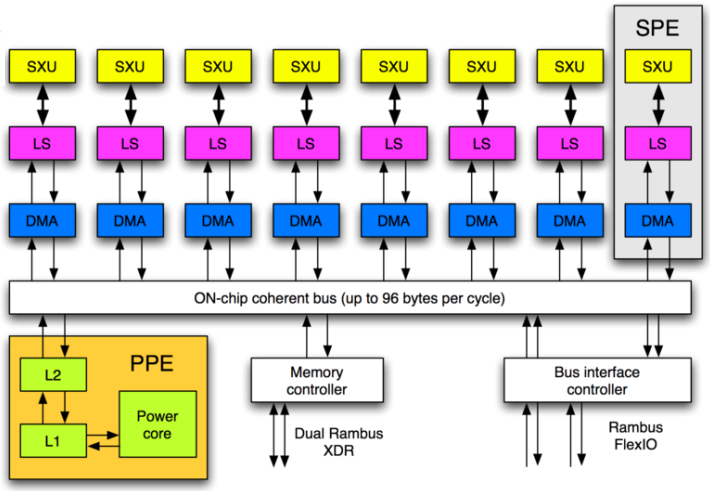
IBM System Technology Group

Examples of Applications Running on Blue Gene

Developed on L, P; many ported to Q

Application	Owner	Application	Owner	Application	Owner
CFD Alya System	Barcelona SC	DFT iGryd	Jülich	BM: SPEC2006, SPEC openmp	SPEC
CFD (Flame) AVBP	CERFACS Consortium	DFT KKRnano	Jülich	BM: NAS Parallel Benchmarks	NASA
CFD dns3D	Argonne National Lab	DFT ls3df	Argonne National Lab	BM: RZG (AIMS,Gadget,GENE, GROMACS,NEMORB,Octopus, Vertex)	RZG
CFD OpenFOAM	SGI	DFT PARATEC	NERSC / LBL	Coulomb Solver - PEPC	Jülich
CFD NEK5000, NEKTAR	Argonne, Brown U	DFT CPMD	IBM/Max Planck	MPI PALLAS	UCB
CFD OVERFLOW	NASA, Boeing	DFT QBOX	LLNL	Mesh AMR	CCSE, LBL
CFD Saturne	EDF	DFT VASP	U Vienna & Duisburg	PETSC	Argonne National Lab
CFD LBM	Erlanger-Nuremberg	Q Chem GAMESS	Ames Lab/Iowa State	MpiBlast-pio Biology	VaTech / ANL
MD Amber	UCSF	Nuclear Physics GFMC	Argonne National Lab	RTM – Seismic Imaging	ENI
MD Dalton	Univ Oslo/Argonne	Neutronics SWEEP3D	LANL	Supernova Ia FLASH	Argonne National Lab
MD ddcMD	LLNL	QCD CPS	Columbia U/IBM	Ocean HYCOM	NOPP / Consortium
MD LAMMPS	Sandia National Labs	QCD MILC	Indiana University	Ocean POP	LANL/ANL/NCAR
MD MP2C	Jülich	Plasma GTC	PPPL	Weather/Climate CAM	NCAR
MD NAMD	UIUC/NCSA	Plasma GYRO (Tokamak)	General Atomics	Weather/Climate Held-Suarez Test	GFDL
MD Rosetta	U Washington	KAUST Stencil Code Gen	KAUST	Climate HOMME	NCAR
DFT GPAW	Argonne National Lab	BM:sppm,raptor,AMG,IRS,sphot	Livermore	Weather/Climate WRF, CM1	NCAR, NCSA

Accelerating Discovery and Innovation in:



- An accelerator based engine (IBM / SONY / TOSHIBA)
- 1 PPE: 2-way SMT PowerPC core, 3.2 Ghz, Up to 32 GB DDR2
- 8 SPEs: 3.2 GHz VMX vector unit, 128 SIMD, 256 Kb SRAM
- 25.6 x p GFLOPS SP / 12.8 x p GFLOPS DP, p = 8, 16, ...
- 25.6 GB/s memory bandwidth

- Asynchronous PPE->SPE / SPE -> SPE communication
- Fast SPE -> SPE communication

Why the CELL Processor ?

- Highest computing power in a single « computing node »
- Fast memory access
- Asynchronysm between data transfers and computation

Issues with the CELL Processor ?

- Data alignment (both for calculations and transfers)
- Heavy use of list DMA
- Small size of the Local Store (SPU local memory)
- Ressources sharing with Dual Cell Based Blade
- Integration into an existing standard framework

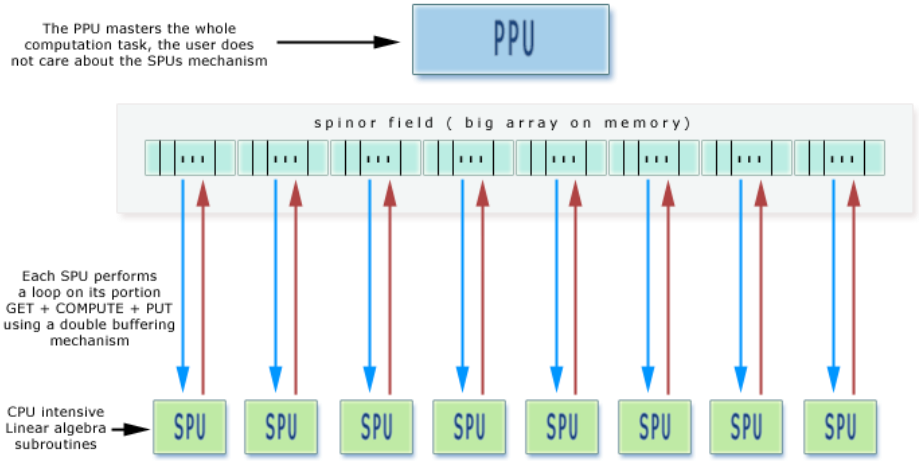
What we have done

- Implementation of each critical kernel on the CELL processor
  - SIMD version of basic operators
  - Appropriate DMA mechanism (efficient list DMA and double buffering)
- Merging of consecutive operations into a unique operator (latency & memory reuse)
- Aggregation of all these implementations into a single and standalone library
  - A single SPU thread holds the whole set of routines
  - SPU thread remains « permanently » active during a working session
- Effective integration into the tmLQCD package
  - Data re-alignment
  - Routine calls replacement (invoke CELL versions in place of native ones)
    This should be the way to commit this back to tmLQCD (external library and « IsCELL » switch)
- Successful tests (QS20 and QS22)

Global organization of the computation

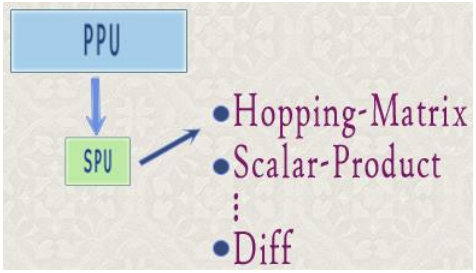
- Task partitioning, distribution, and synchorization are done by the PPU
- Each SPE operates on its data portion by a typical loop of the form

  (DMA get + SIMD Computation + DMA put)



GENERIC LQCD COMPUTATION MECHANISM ON THE CELL

- The SPE, always active, switches to the appropriate operation on each request



Performance results

We consider a $32x16^3$ lattice and CELL-accerated version of tmLQCD using our code (double precision)

QS20			
#SPE	Time(s)	Speedup	GFlops
1	0.109	1.00	0.95
2	0.054	2.00	1.92
3	0.036	3.00	2.89
4	0.027	3.99	3.85
5	0.022	4.98	4.73
6	0.018	5.96	5.78
7	0.015	6.93	6.94
8	0.013	7.88	8.01

QS22			
#SPE	Time(s)	Speedup	GFlops
1	0.0374	1.00	2.76
2	0.0195	1.91	5.31
3	0.0134	2.79	7.76
4	0.0105	3.56	9.90
5	0.0090	4.81	13.27
6	0.0081	5.75	15.87
7	0.0076	6.84	18.88
8	0.0075	7.82	21.59

INTEL i7 quadcore 2.83 Ghz			
Without SSE		With SSE	
1 core	4 cores	1 core	4 cores
0.0820	0.0370	0.040	0.0280

	INTEL i7	CELL (8 SPEs)	
	SSE + 4c	QS20	QS22
GCR (57 iters)	11.05 s	3.78 s	2.04 s
CG (685 iters)	89.54 s	42.25 s	22.78 s

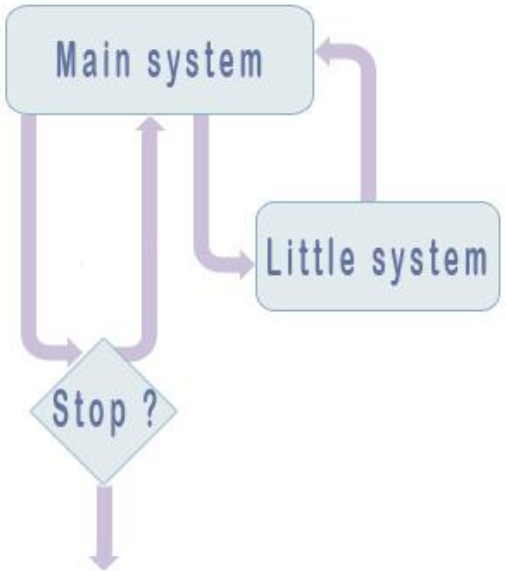
More than **80%** efficiency on a QS20 !!!

ICS-HEART 2010
Epochal Tsukuba, Tsukuba, Japan, June 1, 2010

Preliminary remark about Wilson-Dirac inversion

- We are involved with (very) large-scale linear systems with an implicit principal matrix.
- Because of the implicit form of the Dirac matrix, we only use iterative methods.
- For some parameters (e.g. light masses), the Dirac matrix has a poor condition number.
- One way to tackle ill-conditioned cases is to use the so-called deflation method.
- The deflation method attempts to « isolate » the *low modes* space.
- The deflation method is a *two stages* iterative method.

In order to be efficient, both from the qualitative and the quantitative point of view, the deflation method needs a **good calibration** and lot of **programming efforts**.



TECHNICAL OBSERVATIONS

- The need to approximate the (nearly) kernel subspace.
- Our approximation of the low modes space concerns both the **dimension** and the **basis**.
- An (over/under)dimensioned approximation will lead to degrade the global performance.
- The quality of the «low» eigenvectors approximations is also concerned (although the common belief that this is not important) .
- The second stage subsystem is also solved iteratively.
- The number of iterations of global system and that of the inner system are correlated.

With large gauge configurations and low masses, we have a huge number of iterations and each iteration is costly, thus we are facing a large-scale computation problem.

APPROXIMATION OF THE LOW MODES SUBSPACE (also called DEFLATION SUBSPACE)

- The dimension of the deflation subspace is given as a user parameter (N_s).

The computation of the deflation subspace basis (i.e. N_s approximated eigenvectors) is severely costly.

- We choose to compute each of the N_s vectors as precise as possible (heavy!!!).
- It is expected that the more the deflation subspace is accurate the shorter will be the path to the solution of the global system. This is true in theory, but the reality is versatile.

One argument commonly admitted to pay the price for a good deflation subspace is that this will serve for several inversions.

- Since our **system is large**, we need a way to **extend the deflation basis at a reasonable cost**.
- We choose to **extend the deflation subspace** by means of canonical **block projections**.

BLOCK DECOMPOSITION AND THE LITTLE DIRAC

- The whole lattice divided into blocks $\Theta(\vec{b})$ on a four dimensional grid.
- Every block has a grid coordinate $\vec{b}$ and we have a total number of N_b blocks.
- Assume we have found N_s approximate (global) eigenvectors $\psi_l, l = 1, ..., N_s$.
- We decompose them (by restrictions) onto to the blocks as follows:

$$\phi_l^{\vec{b}}(x) = \begin{cases} \psi_l(x) & \text{if } x \in \Theta(\vec{b}) \,, \\ 0 & \text{otherwise} \,. \end{cases}$$

- We so construct $N_b \cdot N_s$ vectors, which are orthonormalised afterwards.

The *little Dirac operator* A is computed as follows:

$$A_{(\vec{a},k)(\vec{b},l)} = \langle \phi_k^{\vec{a}} | D \phi_l^{\vec{b}} \rangle$$

OUR WORK ON THE BLOCK DECOMPOSITION

We have implemented a generic block decomposition (**multidimensional** and with no restriction on the **number of blocks per axis)** within the tmLQCD package.

For this purpose, we adapted the following modules

- The input parameters list and associated I/O routines.
- The initialization routines.
- The extension of the initial pseudo-eigenvectors using block decomposition.
- The preparation of the little Dirac components.
- The checking routines (to make sure we have good projectors).
- The computation of the little Dirac and its product by a vector.
- MPI communication according to the decomposition.
- Mechanism to store (and later read) the generated intial pseudo-eigenvectors.

EXPERIMENTAL RESULTS

Experimental results of the deflation using tmLQCD

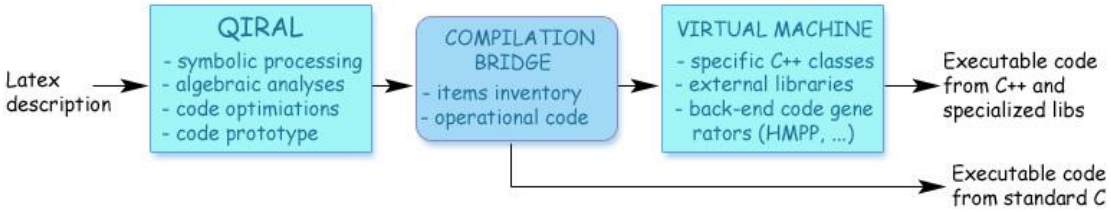
C. Tadonki

Experiments performed with a 64×32^3 lattice (critical kappa = 0.154073) using 128 cores on Jade (cines)
DflFieldIter = 80 DflPolyIter = 20
Get the code
Makefile
Sample input file
Remark: The number of inner iterations need to be carefully examined

N°	kappa	2KappaMu	Method	Block partition	N° of iterations	Time(s)
1	0.1537650	0	cg	-	2424	267
2	0.1537650	0	dflgcr	(1, 1, 2, 2)	46	80
3	0.154073	0.002003	cg	-	783	87
4	0.154073	0.002003	dflgcr	(8, 2, 2, 4)	137	286
5	0.154073	0.002003	dflgcr	(4, 2, 2, 2)	169	299
6	0.154073	0.002003	dflgcr	(2, 2, 2, 4)	198	350
7	0.154073	0.002003	dflgcr	(2, 2, 4, 2)	189	332
8	0.154073	0.002003	dflgcr	(2, 4, 4, 2)	175	323
9	0.154073	0.002003	dflgcr	(2, 2, 2, 2)	214	370
10	0.154073	0.002003	dflgcr	(2, 2, 4, 4)	175	323
11	0.154073	0.002003	dflgcr	(2, 4, 2, 4)	176	328
12	0.154073	0.002003	dflgcr	(4, 2, 2, 4)	153	282
13	0.154073	0.002003	dflgcr	(2, 4, 2, 2)	196	347
14	0.154073	0.002003	dflgcr	(8, 4, 4, 4)	112	481
15	0.154073	0.002003	dflgcr	(8, 2, 4, 2)	137	280
16	0.154073	0.002003	dflgcr	(4, 2, 4, 4)	144	295
17	0.154073	0.002003	dflgcr	(8, 4, 2, 4)	124	328
18	0.154073	0.002003	dflgcr	(2, 4, 4, 4)	166	341
19	0.154073	0.002003	dflgcr	(4, 4, 4, 4)	135	351
20	0.154073	0.002003	dflgcr	(8, 2, 2, 2)	148	272
21	0.154073	0.002003	dflgcr	(4, 4, 2, 4)	143	293
22	0.154073	0.002003	dflgcr	(8, 2, 4, 4)	124	326

Why an Automatic Code Generation System

- As the formulae could be frequently changed or adapted, a push-button system to get the corresponding code is certainly comfortable.
- One application of the Dirac operator involves more than thousand floating point instructions, thus it is hard to plan low level optimization by hand.
- There are different variants of the Dirac operator and several way of expression the calculation depending on the data layout.
- There are different target architectures which can be considered. Thus, generating the code for each of them manually can be tedious and error-prone.
- One way to seek optimal implementation is to filter from an exhaustive search over all possible (or reasonable) .



D. Barthou, G. Grosdidier, M. Kruse, O. Pene and C. Tadonki,
« **QIRAL: A High Level Language for Lattice QCD Code Generation** »,
Programming Language Approaches to Concurrency and Communication-cEntric Software (PLACES'12) in conjunction with the European joint Conference on Theory & Practice of Software (ETAPS), Tallinn, March 24-April 1, 2012.

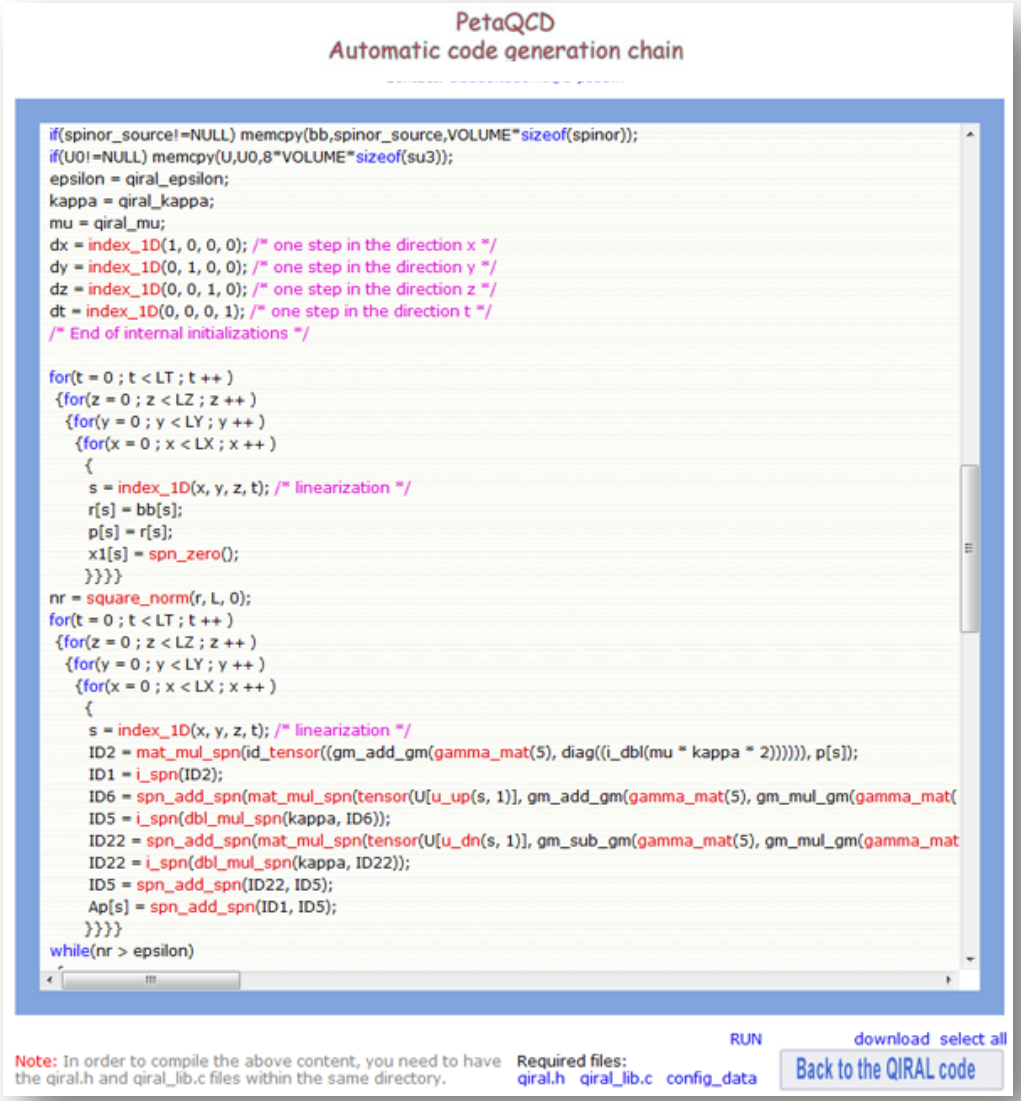
The needs from the pseudo-code to an valid C code

- the variables need to be explicitly declared (QIRAL declares part of the main ones)
- some simplification still need to be done (id(C) x id(S) = id(C x S) and id(n)*u = u)
- special statements need to be appropriately expanded (sum(d in {dx,dy,dz,dt})
- libraries calls are required for macroscopic operations (Ap[L].Ap[L]) ; (Ap[L].r[L])
- specific routines should be called for special operations like (id(S) + gamma(d))*u)
- 4D indexation should be correctly handled/matched with its 1D correspondence
- some profiling and monitoring instructions should be inserted for user convenience
- I/O routines are required for user parameters and data files
- data types need to be correctly captured for further semantic purposes

The last point is particularly important since it will determine the routines to be called (in standard C) or the method execute (C++ / ad-hoc polymorphism).

The C code from the pseudo-code

- The module is based on *Lex* and *Yacc*
- Its generated a valid C code
- The output is associated to an external library for special routines
- A C++ output is planned, will be associated with QDP++ or QUDA through a specific interface
- A web based interface is available



Experimental results

Experimental results of the deflation using tmLQCD

C. Tadonki

Experiments performed with a 64×32^3 lattice (critical kappa = 0.154073) using 128 cores on Jade (cines)
DflFieldIter = 80 DflPolyIter = 20
Get the code
Makefile
Sample input file
Remark: The number of inner iterations need to be carefully examined

N°	kappa	2KappaMu	Method	Block partition	N° of iterations	Time(s)
1	0.1537650	0	cg	-	2424	267
2	0.1537650	0	dflgcr	(1, 1, 2, 2)	46	80
3	0.154073	0.002003	cg	-	783	87
4	0.154073	0.002003	dflgcr	(8, 2, 2, 4)	137	286
5	0.154073	0.002003	dflgcr	(4, 2, 2, 2)	169	299
6	0.154073	0.002003	dflgcr	(2, 2, 2, 4)	198	350
7	0.154073	0.002003	dflgcr	(2, 2, 4, 2)	189	332
8	0.154073	0.002003	dflgcr	(2, 4, 4, 2)	175	323
9	0.154073	0.002003	dflgcr	(2, 2, 2, 2)	214	370
10	0.154073	0.002003	dflgcr	(2, 2, 4, 4)	175	323
11	0.154073	0.002003	dflgcr	(2, 4, 2, 4)	176	328
12	0.154073	0.002003	dflgcr	(4, 2, 2, 4)	153	282
13	0.154073	0.002003	dflgcr	(2, 4, 2, 2)	196	347
14	0.154073	0.002003	dflgcr	(8, 4, 4, 4)	112	481
15	0.154073	0.002003	dflgcr	(8, 2, 4, 2)	137	280
16	0.154073	0.002003	dflgcr	(4, 2, 4, 4)	144	295
17	0.154073	0.002003	dflgcr	(8, 4, 2, 4)	124	328
18	0.154073	0.002003	dflgcr	(2, 4, 4, 4)	166	341
19	0.154073	0.002003	dflgcr	(4, 4, 4, 4)	135	351
20	0.154073	0.002003	dflgcr	(8, 2, 2, 2)	148	272
21	0.154073	0.002003	dflgcr	(4, 4, 2, 4)	143	293
22	0.154073	0.002003	dflgcr	(8, 2, 4, 4)	124	326

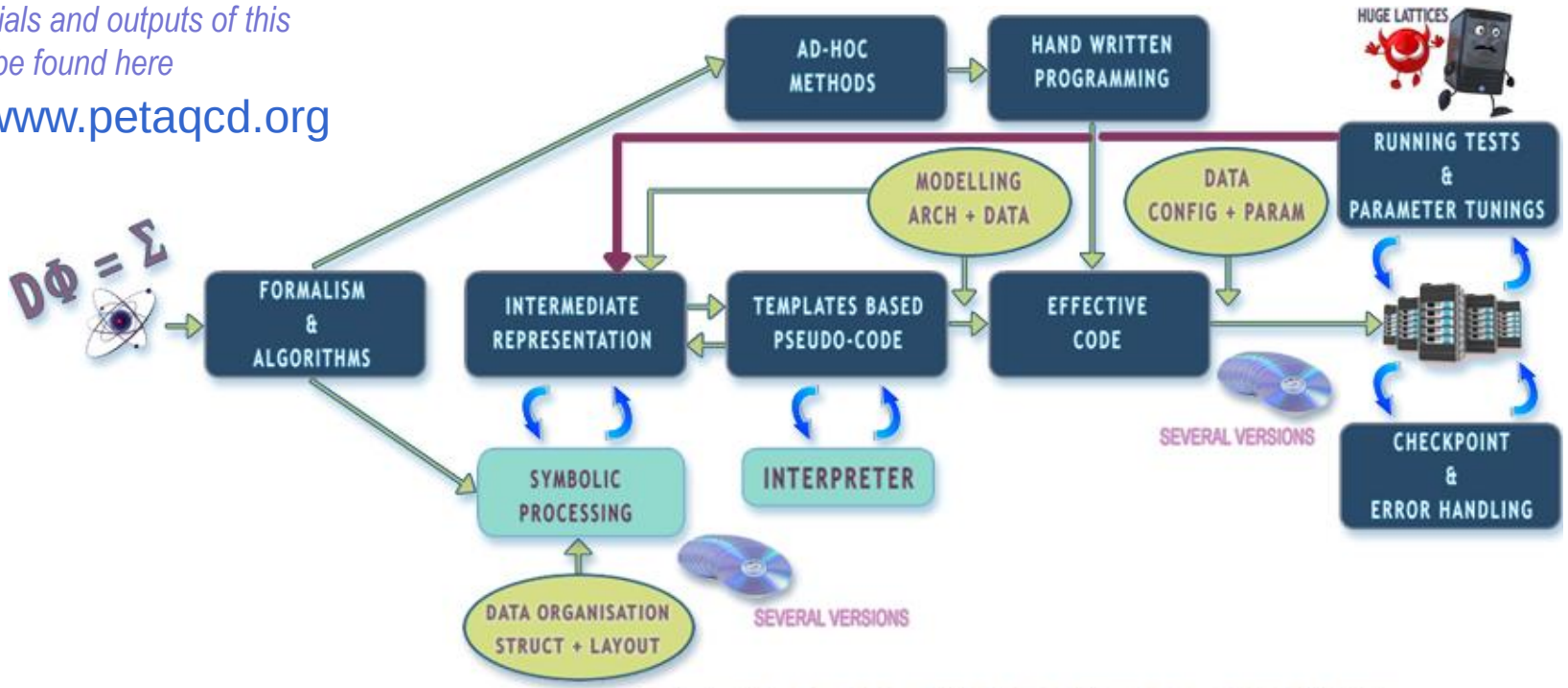
- ANR Project (HPC & Particles Physics)
- Origin of the universe (matter)
- Good cooperative effort
- Multidisciplinary collaboration
- HPC & numerical challenges

National: G. Grosdidier (*coordinator*), P. Roudeau, O. Pène, C. Tadonki, K. Petrov, D. Barthou, M. Kruse, C. Eisenbeis, B. Blossier, L. Morin, F. Bodin, F. Touze, O. Brand-Foissac, J. C. Angles d'Auriac, A. Seznec, A. Cohen, C. Bastoul, and more.

Abroad: K. Urbach, K. Jansen, L. Scorzatto, D. Pleiter, R. Tripiccione

Lot of materials and outputs of this project can be found here

https://www.petaqcd.org



END & QUESTIONS