Towards the Petaflop for Lattice QCD Simulations: the PetaQCD Project

Jean-Christian Anglès d'Auriac[a] <dauriac@lpsc.in2p3.fr>, Denis Barthou[b] <Denis.Barthou@prism.uvsq.fr>, Damir Becirevic[c] <Damir.Becirevic@th.u-psud.fr>, René Bilhaut[d] <bilhaut@lal.in2p3.fr>, François Bodin[k] <francois.bodin@inria.fr>, Philippe Boucaud[c] <Philippe.Boucaud@th.u-psud.fr>, Olivier Brand-Foissac[c] <Olivier.Brand-Foissac@th.u-psud.fr>, Jaume Carbonell[a] <carbonel@lpsc.in2p3.fr>, Christine Eisenbeis[e] <Christine.Eisenbeis@inria.fr>, Pascal Gallard[f] <pascal.gallard@kerlabs.com>, Gilbert Grosdidier[d] <Gilbert.Grosdidier@in2p3.fr>, Pierre Guichon[g] <pierre.guichon@cea.fr>, Pierre-François Honoré[g] <pierre-francois.honore@cea.fr>, Guy Le Meur[d] <lemeur@lal.in2p3.fr>, Olivier Pène[c] <Olivier.Pene@th.u-psud.fr>, Louis Rilling[f] <louis.rilling@kerlabs.com>, Patrick Roudeau[d] <roudeau@lal.in2p3.fr>, André Seznec[h] <seznec@irisa.fr>, Achille Stocchi[d] <stocchi@lal.in2p3.fr>, François Touze[d] <touze@lal.in2p3.fr>

Abstract

The study and design of a very ambitious petaflop cluster exclusively dedicated to Lattice QCD simulations started in early '08 among a consortium of 7 laboratories (IN2P3, CNRS, INRIA, CEA) and 2 SMEs. This consortium received a grant from the French ANR agency in July '08, and the ***PetaQCD*** *project kickoff took place in January '09. Building upon several years of fruitful collaborative studies in this area, the aim of this project is to demonstrate that the simulation of a 256 x 128^3 lattice can be achieved through the HMC/ETMC software, using a machine with efficient speed/cost/reliability/power consumption ratios. It is expected that this machine can be built out of a rather limited number of processors (e.g. between 1000 and 4000), although capable of a sustained petaflop CPU performance.*

The proof-of-concept should be a mock-up cluster built as much as possible with off-the-shelf components, and 2 particularly attractive axis will be mainly investigated, in addition to fast all-purpose multi-core processors: the use of the new brand of IBM-Cell processors (with on-chip accelerators) and the very recent Nvidia GP-GPUs (off-chip co-processors). This cluster will obviously be massively parallel, and heterogeneous. Communication issues between processors, implied by the Physics of the simulation and the lattice partitioning, will certainly be a major key to the project.

1. Framework

Lattice Quantum Chromodynamics (LQCD) is the theory for Nuclear and Sub-nuclear Physics and it simulates the properties of the strong interaction at a sub-nuclear scale, modeling the matter as a quark crystal. The computational demands of Lattice QCD are enormous and have not only played a role in the history of supercomputers but are also helping define their future.

The **PetaQCD** project aims at designing the hardware and software architectures for a sustained performance of over a Petaflop for LQCD simulations with large lattice sizes (up to 128^3 x 256). The **PetaQCD** project is a multi-disciplinary collaboration of teams combining expertise on the Physics of QCD simulation, knowledge of the simulation code HMC (for Hybrid Monte Carlo) already used by the European Twisted Mass Collaboration (ETMC, [Etmc08]), experience on the construction of supercomputers for QCD (through the apeNEXT project) and expertise on optimization techniques for parallel architectures. We propose to optimize the HMC code for LQCD simulation and build a few nodes of a supercomputer mock up able to reach a sustained Petaflop performance for this code.

[a] LPSC, IN2P3/CNRS, 38026 Grenoble, France.
[b] Groupe PRISM, LPRSM, UVSQ, 78035 Versailles, France.
[c] LPT, CNRS, 91405 Orsay, France.
[d] LAL, IN2P3/CNRS, BP34, 91898 Orsay, France.
[k] CAPS-Entreprise, 35000 Rennes, France.
[e] Groupe Alchemy, INRIA Saclay-Ile de France, 91893 Orsay, France.
[f] KERLABS, 35700 Rennes, France.
[g] SPhN, CEA/IRFU, 91191 Gif-sur-Yvette, France.
[h] Groupe ALF, INRIA Rennes-Bretagne Atlantique, 35042 Rennes, France.

Building a special machine for Lattice QCD simulations has been a standard practice. However, the **PetaQCD** project will tackle new scalability issues for Petaflop performance, involving thousands of computing nodes. To achieve this goal, a small structure made of several computing nodes will be studied. Investigations on the data exchange issues between nodes are also essential for the overall performance of the final machine. The study of emerging new architectures (off-chip accelerators such as GPU, on-chip accelerators such as Cell, multi-cores) in the PARA ANR project showed that there is no clear performance/power-consumption/cost overall winner.

A heterogeneous computing node implies multiple code and optimization strategies. In the **PetaQCD** project, we will develop a semi-automatic self-tuning approach for the optimization of performance-critical code fragments of HMC. This approach is similar to what is used in existing self-tuning libraries such as ATLAS or SPIRAL. But it will be the first to be applied to a large application such as HMC and will show the path for performance tuning of heterogeneous supercomputers.

Our current studies show [Ibrahim08] that a 10- to 100-fold performance factor needs to be reached, with currently available hardware, to achieve a sustained Petaflop over a limited number of nodes (about 1000). To achieve this, it will be required to combine a better use of hardware, a thorough QCD algorithm review and finally a better code generation technique.

2. Background and issues

A. Lattice QCD Computation

In Lattice QCD, a four-dimensional space-time continuum is simulated, with quark quantum fields on each lattice site and gluon quantum fields represented by SU(3) matrices on each link between these sites. The calculation aims at computing the average values of physical quantities, which are function of these fields, according to a probability distribution also depending on the fields, and derived by a discretisation procedure from the basic QCD Lagrangian. This average is taken over the full space of all the possible values of the fields. The integration of quark fields is done formally, leading to a complex non-local distribution probability depending only on the gluon fields. We call "gauge configuration" a set of matrices defined on all links. For large lattices the space of gauge configurations is a variety with dimensionality of the order of tenths of billions. Only a Monte-Carlo method allows such a huge calculation. To estimate the average values of the physical quantities we need representative samples of gauge configurations (say about 5000) for every set of parameters, generated according to the above-mentioned probability law. The Hybrid Mont Carlo (HMC) algorithm [Duane87], or variants of it, are used to generate these samples. This is a very heavy calculation. In the following discussion, we will consider an HMC implementation achieved by the ETMC collaboration [Urbach06, Urbach07].

The computation time for Lattice QCD is dominated by few kernels routines. The main kernel routine, called Hopping Matrix, is contributing about 90% of the total execution time [Vranas08]. This routine is responsible for computing the actions of Wilson-Dirac operator. As outlined in Equation (1) below, the actions of Dirac operator involve a sum over quark fields ($\psi_{i+\hat{\mu}}$) multiplied by a gluon gauge link ($U_{i,\mu}$) through the spin projector $(I \pm \gamma_\mu)$.

$$\boxed{\chi_i = \sum_{\mu=\{x,y,z,t\}} \kappa_\mu \left\{ U_{i,\mu} \left(I - \gamma_\mu\right) \psi_{i+\hat{\mu}} + U^\dagger_{i-\hat{\mu},\mu} \left(I + \gamma_\mu\right) \psi_{i-\hat{\mu}} \right\}} \quad (1)$$

The representation of each gauge link is a special unitary SU(3) matrix (3x3 complex variables). The gauge field links go in the four dimensions of the problem space. SU(3) refers to matrices with three colors of quarks that are of special unitary, i.e., unit determinant. The spinors are represented by four SU(3) vectors, each composed of three complex variables.

Building a customary supercomputing facility for simulating Lattice QCD problem has been a commonly used practice [Boyle01, Belletti06]. The motive for building such facilities, with all the associated overheads, is the enormous computational power needed in addition to the special characteristics of the computation of Lattice QCD. This computation tends to have low utilization in most general-purpose computing facilities leading to inefficient power consumption and unrealistic demands on the number of needed computational nodes. For instance to achieve a sustained petaflop scale of computing, we need a million nodes/cores each performing at the one Gflop, what is currently possible for computing Wilson-Dirac operator on Blue-gene/P machines [Doi07]. The efficiency of a computing node does not exceed 20 to 30 percent of the peak performance for Lattice QCD computation.

Previous experiences with building specialized machines show that it is difficult to manage machine above few thousands of nodes because of power requirements and reliability concerns. For instance to build a system of ten thousand nodes, we will need a node capable of 100 Gflops in sustained performance. Effectively, we would like to bridge a performance gap of 100x between the current state-of-art single-node performance and the performance needed to build a reasonably sized system.

The needed scaling in performance is promised by newly emerging technologies for scientific computing such as GPUs and IBM Cell BE. An NVidia GPU G80 is known to have about 300 Gflops peak performance, while synergetic processors inside IBM Cell broadband engine are capable of about 200 Gflops peak performance. These two technologies represent the following two categories of accelerator technologies:

- The former relies on empowering a main general-purpose processor with a specialized accelerator, represented by a CPU assisted by a GPU.
- The latter relies on integrating accelerators with the main processor on chip, providing a heterogeneous system-on-chip kind of architectures, represented by Cell BE.

In the following sections, we present our effort on assessing the potential of using these technologies in the implementation of Lattice QCD main kernel routine to bridge the 100x performance gap.

B. The Use of GPUs for Wilson-Dirac Operator Computation

The use of GPUs for scientific computing is widely under-investigation and gives very promising results for many scientific applications [Owens05]. We investigated the use of NVIDIA G80 GTX GPU board [Nvidia] hosted on a Intel Xeon-based server. This GPU is composed of 16 multiprocessors that are interconnected to banked DRAMs through an impressive bandwidth of 78.6 GB/s.

The parallelization process for GPUs is traditionally done based on vendor compiler. Coding an application is greatly facilitated with the advent of general-purpose programming technology, such as CUDA [Cuda1.1] by NVIDIA. Even though parallelization in done through the compiler, the programmer carries the responsibility of organizing the code (or transforming it) in a way that enable efficient parallelism. A most-do transformation is to remove control-flow instructions whenever possible. For control-flow variables with limited outcomes, lookup tables can be used or redundant computations in conjunction with masking to generate control-flow free code. These techniques are effective in the generation of SIMD operations.

To protect their proprietary hardware, GPU makers allow little control for the application developer. Programmer intervention is needed to explore coding alternatives to determine the best tradeoff between increasing concurrency and increasing resources per thread. In these cases, the application developer should express the problem in multiple alternative representations (task assignments) to the compiler.

In our implementation, we explored the effect of work granularity on performance. The Cuda compiler usually has conflicting constraints on the resources available per block of threads. While it tries to maximize the amount of parallelism (increase the number of threads), it also tries to improve thread's performance by increasing the amount of resources allocated per thread. Certainly, achieving more parallelism, while preserving the performance of a single thread, requires assigning fewer resources per thread. The Cuda compiler tries not to reduce the amount of parallelism below 64 threads, assigning at most 128 registers per thread.

We explored four implementations: the first implementation assigns the computation of a spinor to one thread, called full-spinor version; the second implementation divides a spinor computation into two phases, called half-spinor version; the third implementation is based on dividing a task assigned to a thread of the full-spinor version between sixteen threads; finally, in the fourth implementation eight threads are used for each phase of the half-spinor version. We call the first two implementations as coarse-grained, while the latter ones as fine-grained.

The coarser the thread computation the more stress on the resources. Given that the memory access latency on GPU in the range of 400-600 cycles, it is necessary to reduce the memory access frequency, especially that the caching within the GPU is severely limited in size. For Hopping Matrix computation, we noticed that the less the granularity of the work assigned to a thread the better the performance achieved (28-47% savings in execution time).

For coarse-grained versions, even with the increased memory pressure, the half-spinor version (two threads per spinor computation) provides a better performance compared with the full-spinor (one thread per spinor computation). When the spinor computations are split among 16 threads (fine-granularity) for the half-spinor and the full-spinor techniques, the full-spinor version becomes better than the half-spinor version because the earlier has less frequency of accessing the memory.

The bandwidth of the host CPU memory to the GPU memory has critical impact on performance. In our experiments, the GPU is connected to the host CPU through a PCI Express 16x link with 2.4 GB/s sustained bandwidth (4 GB/s peak performance). The low density of computation compared with exchanged data cause the communication overhead between the GPU and CPU to mount to 40% of the total execution time even for a large lattice size of 32^3x32.

To improve, the ratio of floating point operations to data exchanged, we allocated some of the arrays that are used to hold the intermediate spinors computation to the GPU memory. Using this technique, we reduced the data exchange frequency to one fourth and the contribution of data communication to total execution time is lowered to 11%.

The best performance achieved is about 6.2 Gflops of single precision computations, further details are found in [Ibrahim07a].

C. The Use of Cell BE for Wilson-Dirac Operator Computation

Cell broadband engine (BE) is a unique architecture in integrating specialized accelerator processors, called synergetic processing elements (SPEs), to the main PowerPC based processor. Each SPE has a limited memory, called local store, large register file 128 16-bytes and a specialized SIMD processing element [Hofstee07]. The Cell BE chip integrates XDR memory controller in addition to FlexIO controller. This integration leads to a bandwidth to the memory as high as 25.6 GB/s.

The Cell BE is known for being difficult to program partly because of the detailed control it gives to the programmer over memory management of the different address spaces of SPEs and the main memory. Special DMA calls are usually required to control data transfers. Transforming code to perform efficiently in SIMD mode is an additional traditional obstacle to exploit SPE processors. Relying on compiler is an option that is yet to mature for this kind of architecture.

SIMDizing the code requires aligning the data in way that can be accessed with the least number of data shuffles. Each spinor is accessed in eight different contexts (due to the spin projection operator in Equation 1 above) depending on the space direction. Each access involves different operations and memory access patterns for the real and imaginary part of every complex variable. Unfortunately, we cannot fuse these data optimally at compile time because the same spinor is accessed in eight contexts with different surrounding spinors in each case.

To overcome this difficulty, we devise a technique, called runtime fusion, that fuses the input data used for the computation of multiple consecutive spinors. The real parts of these input data are fused into single register, and similarly for the imaginary part. For instance 16 bytes register requires fusing the computation of two output spinors of double precision or four single precision output spinors. Runtime fusion merges the computation of unrolled loop, thus grouping the data of similar access pattern into 16 bytes word. The results of the computation is then scattered back into multiple output spinors. Cell BE allows such technique because of the large register file.

This technique leads to performing the Hopping Matrix routine with 80 Gflops of single precision computations and 8.7 Gflops of double precision computations. Double precision is not optimized in the current generation of Cell BE, but the promised Cell EDP during the year 2008 should carry an optimized engine for the double precision that is capable of 100 Gflops.

Realistic lattice size needs to be stored in the main memory and be retrieved in pieces for processing. The computational power for single precision Hopping Matrix would require 48 GB/s of the memory, far beyond the 25.6 GB/s bandwidth available.

The input spinors are redundantly accessed 8 times during the computations. Bandwidth can be saved, if non-redundant data are brought to the local store from the external memory, then the redundant part is constructed inside the SPE local store. The saving in bandwidth can be 25% for a sub-lattice size of 16^3x16.

With optimized DMA, the Cell BE can do the computation of the Wilson Dirac operator at 31.2 Gflops in single precision and 8.6 Gflops in double precision. For single precision computation, four SPEs are able to deliver the maximum the chip can afford, while the 8 SPEs doing double precision computation cannot saturate the bandwidth because of their inefficient implementation on Cell BE. In 2008, IBM is expected to release Cell processor with enhanced double precision computations that will be able to deliver at least 16.6 Gflops of double precision computations for the Hopping Matrix routine. Details about this implementation are found in [Ibrahim07a].

D. Summary and Future Projections

Our endeavors in the implementation of the main kernel routine with assist of accelerators, whether on chip like Cell BE [Ibrahim07b] or off-chip such as GPU [Ibrahim07a], show that we can achieve computation at a rate of 8.5 GFlops in double precision (or 31.2 Gflops in single precision).

These performance figures partly bridge the needed 100x gap needed to build a reasonably sized machine. We still need to improve the performance of a node by 12x (about 6x with arrival of Cell eDP, or about 3x for single precision) to build a sustained Petaflop capable machine with about ten thousand nodes. The reason for not fully bridging the performance gap, even with the enormous computational power offered by GPUs and Cell BE, is that the computation to memory access ratio of the Wilson-Dirac computation is low. This ratio ranges from 0.94 to 0.55 flops/byte for double precision, depending on the amount of lattice cached within a computing node.

Considering a Lattice size of $128^3 \times 256$ that can represent heavy-mass quarks system, we will need few thousand of nodes each responsible of a sub-lattice of $16^3 \times 32$. Simulating this sub-lattice size will require 600-700 MB of active dataset per node, a size impossibly cacheable on future architectures for a long time to come.

The reason for choosing a relatively large sub-lattice volume per node is the need to reduce the amount of communicated data relative to amount of computation per node. The computation on a node is proportional to the volume of the sub-lattice, while the communication is proportional to the surface of the sub-lattice.

Obviously, bridging the gap further requires additional effort on the algorithmic aspect of the computation to improve the computation density in addition to better exploit of the advances in the computational power of future computing nodes. We finally assert that improvements in performance are likely to be more correlated with improvements in the bandwidth to memory of computing nodes.

3. Methodology

LQCD is already an ambitious challenge in itself. The enormous computing power that it requires has driven research in high performance computing since it has lead for more than 20 years to the conception and realization of dedicated computers, the APE series up to the apeNEXT, and the Columbia series up to the QCDOC which became the root of the BlueGene computers. The basic starting point of **PetaQCD** is that even though promises in speed increase are eventually met in the next 3-5 years, analysis of the current LQCD code shows that we will still miss one to two orders of magnitude in the performance to achieve the sustained Petaflops at that time. There is currently no clear hint that hardware technology upgrades alone could achieve the required performance.

Therefore radical breakthroughs are required and it is not yet known where they can be found. There is not a single road for possible improvements. Improvements can be gained from the physics model and numerical scheme of LQCD, or in software compilers, or in architecture improvement, most probably all of them. Moreover, myriads of processors will obviously be required for this performance and we therefore will have to face the hardware reliability issue.

It is clear that addressing these issues in a general framework would be out of reach. Restricting our project to LQCD makes our objectives reachable by tight cooperation in the whole computation chain, from algorithms down to hardware. We also expect that any small improvement we will achieve can be in turn exploited in more general contexts. For instance new software optimizations can be integrated in our semi-automatic tools, while addressing uncertainty issues can open new insights in numerical analysis.

In our previous joint work between several participants (LPT, IRISA, INRIA, PRISM & LAL) of **PetaQCD** we have already identified the main bottleneck, namely the communication bottleneck. The ratio between computations and communications on the current QCD code is roughly 5 floating operations per data input. Even though this ratio is not too bad, it is a constant and will obviously not match the expected latency ratio (about 500 clock cycles for a data transfer instead a few clock cycles for a floating point operation, expected to grow), neither the expected bandwidth for communications. Therefore we have identified the **communication** architecture as one of the main issues. The second one is **scalability**, how to manage larger lattices, thousands of processors, larger power consumption, larger combinatorial of the problem of mapping code on architecture. The last issue is related to the **precision** of computations regarding scalability, possible transient errors, trade-off of precision versus performance.

For tackling these three intrinsically related issues, communication, scalability, and precision, we propose different – also interleaved – roads.

A. Hierarchy and multilevel parallelism

First, we propose to consider **hierarchy** at all levels: hierarchy in the network architecture as well as in the algorithmic scheme. This includes heterogeneity in the computation model, possible SIMD instructions at the lowest level, vector coprocessors, multi-cores, clusters.

At the software level, we have to model the communication scheme and provide code for its management. This includes managing registers, vector registers, local memories, cache memories, and communications at all levels. In the algorithmic part, that means reconsidering different schemes for code and lattice (possibly hierarchical) partitioning. But this also requires reconsidering the associated preconditioning scheme and eventually physics validation.

B. Combined solution space exploration

Then, we will use systematic **space exploration** for this algorithm/software/processor architecture co-design problem. At the hardware level, that means having a systematic and robust scheme for the processors and network model of performance. It will be based on today and short term expected numbers and will be validated on different actual processors: general processors, GPU coprocessors, and IBM Cell processors. The scalability issue will be validated by experiments on existing massively parallel processors (a BlueGene/P computer, and a standard multi-core cluster). At the software level, this requires using, adapting and improving our semi-automatic tools for the generation of different code versions, from the lowest level (instruction scheduling, register allocation) up to highest levels (local memories, MPI communications). Since the combinatorial of exhaustive search is very high, random or now classical machine learning or genetic algorithms will be used. At the algorithm level, we will consider alternative numerical schemes. Matching the schemes to the provided architecture will most probably not be predictable statically and we plan to consider also online (at run time) choices of preconditioning schemes as well as alternative partitioning schemes. Changing the scheme alters also the convergence, this will be also systematically estimated by a validation of the physics result as well as the consequence in terms of overall execution time.

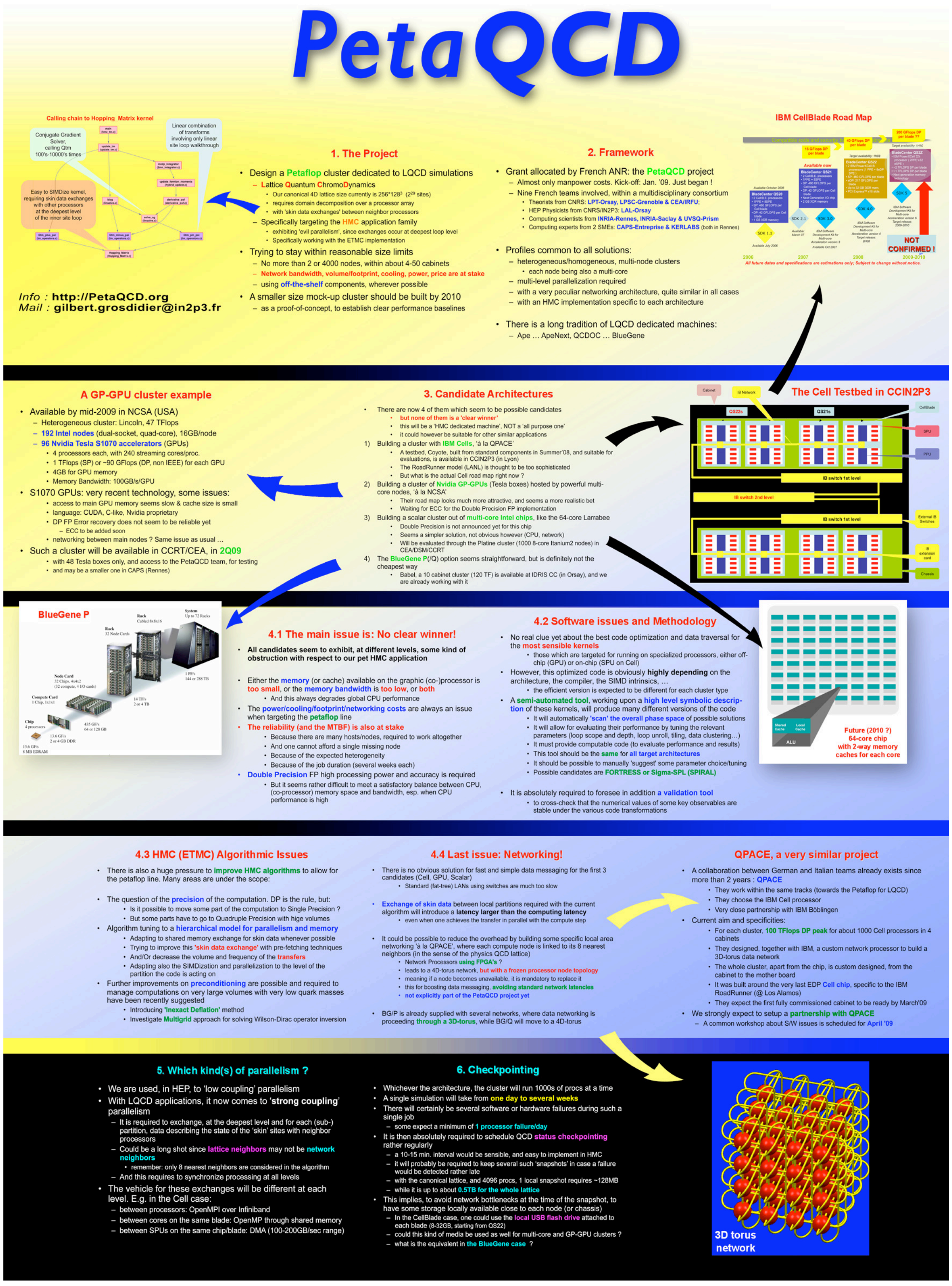
C. Relaxing synchronization constraints

Last, gaining this factor of 10 to 100 requires a radical change in the computation paradigm. Especially the main communication bottleneck must be addressed. We propose to consider **relaxing** usual requirements of **deterministic and reproducible** computations. This means that we will consider different precisions levels at different steps of the computation. For avoiding costly communications we will consider synchronizing different parts of the algorithms less often than required, or even avoiding costly synchronization barriers by **asynchronous** - and by essence non predictable - communications. More classically, since computations are cheaper than communications, it is likely that some computations units are idle and they can be exploited for control tasks such as error detection or redundant computations, or even races between redundant computations. As mentioned in §3.B, we will also consider auto-tuning of the code at run time, meaning different versions of the code or algorithms, this means also different communications schemes. This 3rd part is the most risky part of **PetaQCD** but each of these "non-standard" roads will be systematically validated by the final physics result.

We choose to structure our **PetaQCD** project around three working groups (WG). The 1st one is related to hardware model and software tools, the 2nd one to the physics and algorithmic part, and the 3rd one to the specific problem of fault tolerance. Every partner is concerned by these three packages as these WG involve algorithmic, software and architectural consideration.

To the best of our knowledge, this is the first time that such a project involving physicists and computer scientists is addressed with respect to the most recent results and trends in high performance computing. Related software environments for code generation, especially for specific co-processors and management of local memories, and exhaustive search, are only starting to emerge. The fault tolerance issue has seldom been addressed in scientific computing. Lastly gaining this already mentioned order of magnitude is a quite challenging question that can only be reached by questioning all levels of this chain of physics, algorithms, software (systems and programming environments) and hardware levels.

The ultimate aim is to provide at the end of the project, instead of a single solution (algorithmic/ code/ hardware), a range of possible solutions, together with search tools that can be quickly adapted to the state of the technology at that time.



PetaQCD

Calling chain to Hopping_Matrix kernel

Conjugate Gradient Solver, calling Qtm 100's-10000's times

Linear combination of transforms involving only linear site loop walkthrough

Easy to SIMDize kernel, requiring skin data exchanges with other processors at the deepest level of the inner site loop

Info : **http://PetaQCD.org**
Mail : **gilbert.grosdidier@in2p3.fr**

1. The Project

- Design a **Petaflop** cluster dedicated to LQCD simulations
  - Lattice Quantum ChromoDynamics
    - Our canonical 4D lattice size currently is 256*128³ (2²⁹ sites)
    - requires domain decomposition over a processor array
    - with 'skin data exchanges' between neighbor processors
  - Specifically targeting the HMC application family
    - exhibiting 'evil parallelism', since exchanges occur at deepest loop level
    - Specifically working with the ETMC implementation
- Trying to stay within reasonable size limits
  - No more than 2 or 4000 nodes, within about 4-50 cabinets
  - **Network bandwidth, volume/footprint, cooling, power, price are at stake**
  - using **off-the-shelf** components, wherever possible
- A smaller size mock-up cluster should be built by 2010
  - as a proof-of-concept, to establish clear performance baselines

2. Framework

- Grant allocated by French ANR: the **PetaQCD** project
  - Almost only manpower costs. Kick-off: Jan. '09. Just began !
  - Nine French teams involved, within a multidisciplinary consortium
    - Theorists from CNRS: **LPT-Orsay, LPSC-Grenoble & CEA/IRFU**;
    - HEP Physicists from CNRS/IN2P3: **LAL-Orsay**
    - Computing scientists from **INRIA-Rennes, INRIA-Saclay & UVSQ-Prism**
    - Computing experts from 2 SMEs: **CAPS-Entreprise & KERLABS** (both in Rennes)
- Profiles common to all solutions:
  - heterogeneous/homogeneous, multi-node clusters
    - each node being also a multi-core
  - multi-level parallelization required
  - with a very peculiar networking architecture, quite similar in all cases
  - with an HMC implementation specific to each architecture
- There is a long tradition of LQCD dedicated machines:
  - Ape … ApeNext, QCDOC … BlueGene

IBM CellBlade Road Map

All future dates and specifications are estimations only; Subject to change without notice.

A GP-GPU cluster example

- Available by mid-2009 in NCSA (USA)
  - Heterogeneous cluster: Lincoln, 47 TFlops
  - **192 Intel nodes** (dual-socket, quad-core), 16GB/node
  - **96 Nvidia Tesla S1070 accelerators** (GPUs)
    - 4 processors each, with 240 streaming cores/proc.
    - 1 TFlops (SP) or ~90 GFlops (DP, non IEEE) for each GPU
    - 4GB for GPU memory
    - Memory Bandwidth: 100GB/s/GPU
- S1070 GPUs: very recent technology, some issues:
  - access to main GPU memory seems slow & cache size is small
  - language: CUDA, C-like, Nvidia proprietary
  - DP FP Error recovery does not seem to be reliable yet
    - ECC to be added soon
  - networking between main nodes ? Same issue as usual …
- Such a cluster will be available in CCRT/CEA, in **2Q09**
  - with 48 Tesla boxes only, and access to the PetaQCD team, for testing
  - and may be a smaller one in CAPS (Rennes)

3. Candidate Architectures

- There are now 4 of them which seem to be possible candidates
  - **but none of them is a 'clear winner'**
  - this will be a 'HMC dedicated machine', NOT a 'all purpose one'
  - it could however be suitable for other similar applications

1) Building a cluster with **IBM Cells**, 'à la QPACE'
   - A testbed, Coyote, built from standard components in Summer'08, and suitable for evaluations, is available in CCIN2P3 (in Lyon)
   - The RoadRunner model (LANL) is thought to be too sophisticated
   - But what is the actual Cell road map right now ?
2) Building a cluster of **Nvidia GP-GPUs** (Tesla boxes) hosted by powerful multi-core nodes, 'à la NCSA'
   - Their road map looks much more attractive, and seems a more realistic bet
   - Waiting for ECC for the Double Precision FP implementation
3) Building a scalar cluster out of **multi-core Intel chips**, like the 64-core Larrabee
   - Double Precision is not announced yet for this chip
   - Seems a simpler solution, not obvious however (CPU, network)
   - Will be evaluated through the Platine cluster (1000 8-core Itanium2 nodes) in CEA/DSM/CCRT
4) The **BlueGene P**(/Q) option seems straightforward, but is definitely not the cheapest way
   - Babel, a 10 cabinet cluster (120 TF) is available at IDRIS CC (in Orsay), and we are already working with it

The Cell Testbed in CCIN2P3

BlueGene P

4.1 The main issue is: No clear winner!

- **All candidates seem to exhibit, at different levels, some kind of obstruction with respect to our pet HMC application**
- Either the **memory** (or cache) available on the graphic (co-)processor is **too small**, or the **memory bandwidth** is **too low**, or **both**
  - And this always degrades global CPU performance
- The **power/cooling/footprint/networking costs** are always an issue when targeting the **petaflop** line
- **The reliability (and the MTBF) is also at stake**
  - Because there are many hosts/nodes, required to work altogether
  - And one cannot afford a single missing node
  - Because of the expected heterogeneity
  - Because of the job duration (several weeks each)
- **Double Precision** FP high processing power and accuracy is required
  - But it seems rather difficult to meet a satisfactory balance between CPU, (co-processor) memory space and bandwidth, esp. when CPU performance is high

4.2 Software issues and Methodology

- No real clue yet about the best code optimization and data traversal for the **most sensible kernels**
  - those which are targeted for running on specialized processors, either off-chip (GPU) or on-chip (SPU on Cell)
- However, this optimized code is obviously **highly depending** on the architecture, the compiler, the SIMD intrinsics, …
  - the efficient version is expected to be different for each cluster type
- A **semi-automated tool**, working upon a **high level symbolic description** of these kernels, will produce many different versions of the code
  - It will automatically **'scan' the overall phase space** of possible solutions
  - It will allow for evaluating their performance by tuning the relevant parameters (loop scope and depth, loop unroll, tiling, data clustering…)
  - It must provide computable code (to evaluate performance and results)
  - This tool should be the **same** for **all target architectures**
  - It should be possible to manually 'suggest' some parameter choice/tuning
  - Possible candidates are **FORTRESS or Sigma-SPL (SPIRAL)**
- It is absolutely required to foresee in addition **a validation tool**
  - to cross-check that the numerical values of some key observables are stable under the various code transformations

Future (2010 ?) 64-core chip with 2-way memory caches for each core

4.3 HMC (ETMC) Algorithmic Issues

- There is also a huge pressure to **improve HMC algorithms** to allow for the petaflop line. Many areas are under the scope:
- The question of the **precision** of the computation. DP is the rule, but:
  - Is it possible to move some part of the computation to Single Precision ?
  - But some parts have to go to Quadruple Precision with hige volumes
- Algorithm tuning to a **hierarchical model for parallelism and memory**
  - Adapting to shared memory exchange for skin data whenever possible
  - Trying to improve this **'skin data exchange'** with pre-fetching techniques
  - And/Or decrease the volume and frequency of the **transfers**
  - Adapting also the SIMDization and parallelization to the level of the partition the code is acting on
- Further improvements on **preconditioning** are possible and required to manage computations on very large volumes with very low quark masses have been recently suggested
  - Introducing **'Inexact Deflation'** method
  - Investigate **Multigrid** approach for solving Wilson-Dirac operator inversion

4.4 Last issue: Networking!

- There is no obvious solution for fast and simple data messaging for the first 3 candidates (Cell, GPU, Scalar)
  - Standard (fat-tree) LANs using switches are much too slow
- **Exchange of skin data** between local partitions required with the current algorithm will introduce a **latency larger than the computing latency**
  - even when one achieves the transfer in parallel with the compute step
- It could be possible to reduce the overhead by building some specific local area networking 'à la QPACE', where each compute node is linked to its 8 nearest neighbors (in the sense of the physics QCD lattice)
  - Network Processors **using FPGA's** ?
  - leads to a 4D-torus network, **but with a frozen processor node topology**
  - meaning if a node becomes unavailable, it is mandatory to replace it
  - this for boosting data messaging, **avoiding standard network latencies**
  - **not explicitly part of the PetaQCD project yet**
- BG/P is already supplied with several networks, where data networking is proceeding **through a 3D-torus**, while BG/Q will move to a 4D-torus

QPACE, a very similar project

- A collaboration between German and Italian teams already exists since more than 2 years : **QPACE**
  - They work within the same tracks (towards the Petaflop for LQCD)
  - They choose the IBM Cell processor
  - Very close partnership with IBM Böblingen
- Current aim and specificities:
  - For each cluster, **100 TFlops DP peak** for about 1000 Cell processors in 4 cabinets
  - They designed, together with IBM, a custom network processor to build a 3D-torus data network
  - The whole cluster, apart from the chip, is custom designed, from the cabinet to the mother board
  - It was built around the very last EDP **Cell chip**, specific to the IBM RoadRunner (@ Los Alamos)
  - They expect the first fully commissioned cabinet to be ready by March'09
- We strongly expect to setup a **partnership with QPACE**
  - A common workshop about S/W issues is scheduled for **April '09**

5. Which kind(s) of parallelism ?

- We are used, in HEP, to 'low coupling' parallelism
- With LQCD applications, it now comes to '**strong coupling**' parallelism
  - It is required to exchange, at the deepest level and for each (sub-) partition, data describing the state of the 'skin' sites with neighbor processors
  - Could be a long shot since **lattice neighbors** may not be **network neighbors**
    - remember: only 8 nearest neighbors are considered in the algorithm
  - And this requires to synchronize processing at all levels
- The vehicle for these exchanges will be different at each level. E.g. in the Cell case:
  - between processors: OpenMPI over Infiniband
  - between cores on the same blade: OpenMP through shared memory
  - between SPUs on the same chip/blade: DMA (100-200GB/sec range)

6. Checkpointing

- Whichever the architecture, the cluster will run 1000s of procs at a time
- A single simulation will take from **one day to several weeks**
- There will certainly be several software or hardware failures during such a single job
  - some expect a minimum of **1 processor failure/day**
- It is then absolutely required to schedule QCD **status checkpointing** rather regularly
  - a 10-15 min. interval would be sensible, and easy to implement in HMC
  - it will probably be required to keep several such 'snapshots' in case a failure would be detected rather late
  - with the canonical lattice, and 4096 procs, 1 local snapshot requires ~128MB
  - while it is up to about **0.5TB for the whole lattice**
- This implies, to avoid network bottlenecks at the time of the snapshot, to have some storage locally available close to each node (or chassis)
  - In the CellBlade case, one could use the **local USB flash drive** attached to each blade (8-32GB, starting from QS22)
  - could this kind of media be used as well for multi-core and GP-GPU clusters ?
  - what is the equivalent in **the BlueGene case** ?

Figure 1: The PetaQCD poster

4. What is going on elsewhere in this field?

Lattice QCD is the only approach that allows studying, in a systematic way, the strong interaction between fundamental constituents of matter. It is of great importance in three main areas: to check the Standard Model of particle physics and possibly discover its limits; to understand the internal structure of nucleons; to determine the properties of strongly interacting matter under extreme conditions. Results from LQCD are needed to extract the physics content of measurements obtained at present and future experimental facilities running (or planned) in Europe, USA or Japan. The accuracy of these results has to be matched with the experimental accuracy and this dictates the needed computing power to be allocated to these studies.

The main efforts from the LQCD community can be illustrated by the present sustained computing capabilities[1] (in Tflops, cf [Fnal07]):

Germany	10-15
Italy + France	5 + 0.5
Japan	14-18
United-Kingdom	4-5
USA	11

Currently, most of the computing power is provided by dedicated computers, specifically developed for LQCD. As an example, there are three QCDOC computers, each with three sustained TFlops installed in USA and Scotland in 2005. ApeNEXT computers have been also installed in Italy and Germany.

To achieve the physics goals, which consist in having a percent accuracy on several quantities of interest, all groups agree that computing capabilities of about one Petaflop are needed.

The US QCD community has developed coherent plans on software and hardware developments since year 2000 with two five-year plans. The SciDAC initiative permits the installation of two QCDOC machines and eight commodity clusters of moderate size (<0.5 Tflops each). During the current grant larger commodity clusters are being installed and people are working on the successor to the QCDOC which is expected to be a petascale machine at 0.01$/megaflops/s. Grants are also provided by the High Energy and Nuclear Physics community through the LQCD project started in 2006. They plan to achieve 18 Tflops sustained in 2008-2009 for a total cost of 9 M$. The use of super-computers (financed by the DOE) is also expected to provide a similar amount of computing power for LQCD.

This short summary indicates that efforts are on the way in the US to increase the computing capabilities devoted to LQCD and financed by the high energy and nuclear physics communities. In parallel they continue developments to obtain better cost-effective dedicated machines. In Europe, we are aware of hardware developments lead by the **QPACE** project to build a computer prototype for LQCD, based on the IBM Cell, towards a petascale machine [Qpace08].

Within 10-15 years, it will be possible to integrate on a single chip a huge memory in the gigabyte range and 100s and may be 1,000s of processing elements. On the early design phase of a processor, one has to dimension various parameters such as memory hierarchy or processing units. Such design space exploration was traditionally done through instruction level simulations. With the advent of moderately parallel single chip component, the same methodology is still in use. New methodologies will be required for chips featuring 100s and may be 1000s of processing elements. Designers will face the challenging problem of choosing the global structure for the multi-core. In particular due to memory access time constraints, the on-chip memory will be organized as a 'memory hierarchy' featuring several levels. The effective performance of the overall system depends on the correct dimensioning of all the parameters in the memory hierarchy: in practice, if a single parameter is underestimated the overall system performance will be limited by this parameter.

The exploration of the design space for dimensioning the memory hierarchy of 1,000s cores architecture requires new methodology. Design space exploration has already been widely used in the domain of embedded processors (for instance in the PICO codesign framework). Other works target general multicore architectures. We propose a new methodology that will mix analytical models and simulation in order to explore the design space of memory hierarchy of massively parallel multi-cores.

This hardware exploration will be done in tight connection with the exploration of the space of different code versions. Iterative compilation and code generation is more and more widely used as a solution to the complexity of processors (Oceans european project, Atlas, Spiral, FFTW, work of Alchemy, Caps and Prism).

[1] The French community has contributed to the Rome computing centre installed in La Sapienza University

We will perform the iterative search starting from the current HMC code as well as the equational formulation. We will also add a new degree of freedom by exploring alternatives algorithms for LQCD.

Acknowledgements

This work is supported by French National Research Agency (ANR) through COSINUS program (project PETAQCD n°ANR- 08-COSI-010).

References

[Belletti06] F. Belletti et al., Computing for LQCD: apeNEXT. *Computing in Science and Engineering*, 8(1):18-29, 2006.

[Boyle01] P. A. Boyle et al., Status of the QCDOC project. *arXiv:hep-lat/0110124v1*, 2001.

[Doi07] J. Doi. Performance evaluation and tuning of lattice QCD on the next generation Blue Gene. *Proceeding of Science*, Oct 2007.

[Duane87] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, Hybrid Monte Carlo. *Phys. Lett., B195*:216-222, 1987.

[Etmc08] R. Baron et al., Status of ETMC simulations with Nf=2+1+1 twisted mass fermions, *XXVI International Symposium on Lattice Field Theory*, July 14-19, 2008, Williamsburg, Virginia, USA.

[Fnal07] From the white paper "Computational resources for Lattice QCD 2010-2014" http://theory.fnal.gov/theorybreakout2007/LatticeQCD2010-2014.pdf.

[Hofstee07] H.P. Hofstee et al., Cell Broadband Engine Technology and Systems, *IBM J. Research and Development*, vol. 51, no. 5, 2007.

[Ibrahim07a] K. Z. Ibrahim, F. Bodin, and O. Pene. Fine-grained Parallelization of Lattice QCD Kernel Routine on GPUs. *First Workshop on General Purpose Processing on Graphics Processing Units*, Northeastern Univ., Boston, Oct 2007.

[Ibrahim07b] K. Z. Ibrahim and F. Bodin. Implementing Wilson-Dirac Operator on the Cell Broadband Engine. *INRIA/IRISA Tech. report PI 1880*, Dec. 2007.

[Ibrahim08] K. Z. Ibrahim et al., Simulation of the Lattice QCD and Technological Trends in Computation, [arXiv:0808.0391], *Submitted to the 14th International Workshop on Compilers for Parallel Computers*, Aug 2008

[Nvidia] NVIDIA Corporation. http://www.nvidia.com/.

[Cuda1.1] NVIDIA Cuda 1.1. http://developer.nvidia.com/object/cuda.html, 2008.

[Owens05] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell. A Survey of General-Purpose Computation on Graphics Hardware. *Eurographics 2005, State of the Art Reports*, pages 21-51, Aug 2005.

[Qpace08] G. Goldrian et al., QPACE: Quantum Chromodynamics Parallel Computing on the Cell Broadband Engine, *Computing in Science and Engineering*, vol. 10, no. 6, Nov./Dec. 2008

[Urbach07] C. Urbach. Lattice QCD with Two Light Wilson Quarks and Maximal Twist. *The XXV International Symposium on Lattice Field Theory*, 2007.

[Urbach06] C. Urbach, K. Jansen, A. Shindler, and U. Wenger. HMC Algorithm with Multiple Time Scale Integration and Mass Preconditioning. *Computer Physics Communications*, 174:87, 2006.

[Vranas08] P. Vranas et al., Massively Parallel Quantum Chromodynamics. *IBM journal of research and development*, 52(1/2), 2008.