Parallel Quasi-Monte Carlo Simulation of Ultrafast Carrier Transport

Aneta Karaivanova
(Joint work with E. Atanassov and T. Gurov)
Institute of Information and Communication Technologies
Bulgarian Academy of Science
anet@parallel.bas.bg

IICT-Centre for Advanced Computing

Strategic targets:

Sustainable development of the institute as a national leader in the information and communication technologies, with internationally visible and recognized results.

Mission:

To perform basic and applied research in the fields of computer science and information and communication technologies, as well as to develop interdisciplinary innovations.

Research staff – 106 (9 Full Professors, 50 Assoc. Professors, 47 Assistant Professor), 50 PHD students

NATIONAL e-Infrastructure responsibilities of IICT:

- ➢ IICT is the **National Centre for HPC and Distributed Computing** (since July 2014, Bulgarian Roadmap on RIs)
  - ✓ **A new state-of-the-art computing system with more than 400 TFs** will be available soon (in 2 months)
- ➢ IICT coordinates consortium of 3 universities and 3 institutes in the **Center of Excellence "Supercomputer Applications"**
- ➢ IICT **coordinates the National Grid Initiative** (NGI) and presents it in the EGI.eu Council since 2010.
- ➢ IICT **hosts the main node of BREN** and is a **member of the Board** of Bulgarian Research and Educational Network (BREN).
- ➢ IICT-BAS is responsible for the operations of the *Bulgarian Academic Certification Authority (*http://ca.acad.bg/*)* which is authorized to issue digital Grid certificates free of charge for all Bulgarian Grid users and Grid hosts

Departments

- Computer Networks and Architectures
- Parallel Algorithms
- Scientific Computations
- Mathematical Methods for Sensor Data Processing
- Linguistic Modelling
- Information Technologies for Security
- Grid Technologies and Applications
- Technologies for Knowledge Management and Processing
- Modelling and Optimization
- Signal Processing and Pattern Recognition
- Information Processes and Decision Support Systems
- Intelligent systems
- Embedded Intelligent Technologies
- Communication Systems and Services
- Hierarchical Systems

Structure of the R&D activities

The research and development activities of IICT during 2014 are performed into the framework of the 71 main projects:

- 15 funded by the budget subsidiary
- 16 supported by the Bulgarian Science Fund (BSF)
- 15 funded by the Operational Programs: 13 by OP „Development of the Competitiveness of the Bulgarian Economics“ and 2 by OP „Human Resources Development”
- 17 international projects: 14 funded by EC
- 11 R&D contracts directly with industrial enterprise

Just awarded a new EU project: **Centre of Excellence in Mathematical Modeling and Advanced Computing**

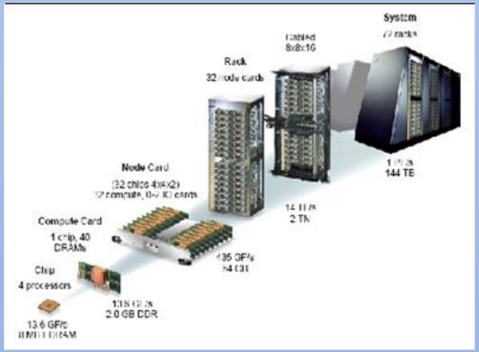
ИИКТ

Center of Excellence on Supercomputing Applications: SuperCA++,BSF Grant DCVP 02/1

Consortium: IICT – BAS (coordinator), SU, TU – Sofia, MU – Sofia, IM – BAS, NIGGG - BAS

Infrastructure: Supercomputer IBM Blue Gene/P at NSCC, HPC Cluster at IICT – BAS

The project creates a critical mass of highly qualified scientists. The core team consists of more than 80 people: about 56% of them are PhD students and young researchers.



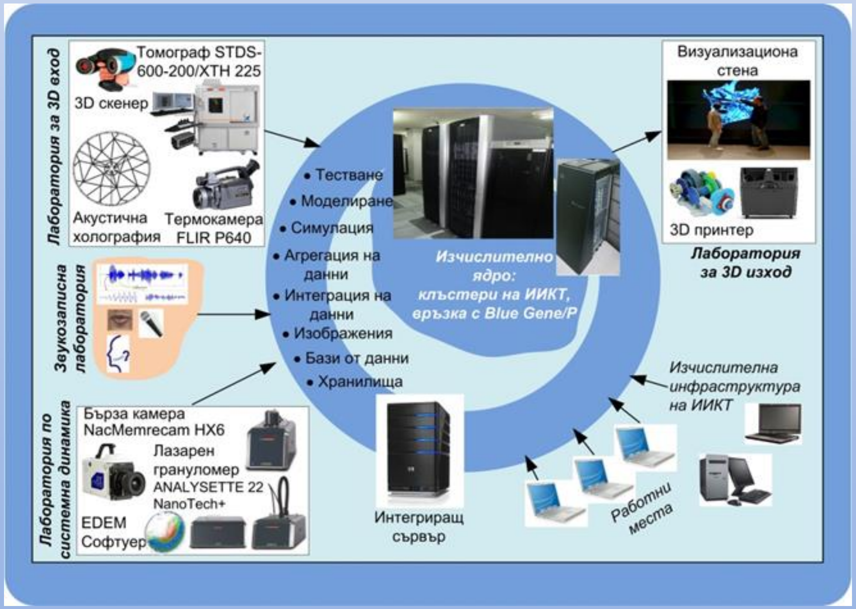
Advanced Computing for Innovations: AComIn, FP7-REGPOT-2012-2013-1, GA 316087

Major Objectives:

•Strengthening the human potential

•Setting up Smart Periphery Lab

•Organization and training of user communities



Outline of this talk

- Introduction
- Monte Carlo, quasi-Monte Carlo and hybrid approach
- MPI implementation
- Bulgarian HPC resources
- Scalability study
- Numerical and timing results on Blue Gene/P and HPC cluster
- GPU-based implementation
- Conclusions and future work

Introduction

- The problem of ***stochastic modeling of electron transport*** has high theoretical and practical importance
- Stochastic numerical methods (***Monte Carlo methods***) are based on simulation of random variables/processes and estimation of their statistical properties. They have some advantages for high dimensional problems, problems in complicated domains or when we are interested in part of the solution.
- ***Quasi-Monte Carlo methods*** are deterministic methods which use low discrepancy sequences. For some problems they offer higher precision and faster convergence.
- ***Randomized quasi-Monte Carlo methods*** use randomized (scrambled) quasirandom sequences. They combine the advantages of Monte Carlo and quasi-Monte Carlo.
- The problems are highly computationally intensive. Here we present scalability results for various HPC systems.

Monte Carlo Methods

- J is a quantity to be estimated via a MCM
- Θ is a random variable with E[Θ] = J
- Θ_N is the estimator with N samples
- The MCM convergence rate is $\mathbf{N^{-1/2}}$ with sample size N ($\boldsymbol{\varepsilon \approx \sigma(\theta) N^{-1/2}}$);
  - Probabilistic result – there is no absolute upper bound.
  - The statistical distribution of the error is a normal random variable.
- The MCM error and the sample size are connected by:

  $\varepsilon = O(\sigma N^{-1/2})$, $N = O(\sigma/\varepsilon)^2$
- The computing time is proportional to N, i.e., it increases very fast if a better accuracy is needed.
- How to increase the convergence:
  - Variance reduction
  - Change of the underlying sequence
- In this talk we consider improvement through sequence optimization

Low discrepancy (quasirandom) sequences

- The quasirandom sequences are deterministic sequences constructed to be as uniformly distributed as mathematically possible (and, as a consequence, to ensure better convergence for the integration)
- The uniformity is measured in terms of discrepancy which is defined in the following way: For a sequence with N points in $[0,1]^s$ define

$$\mathbf{R_N(J) = 1/N\#\{x_n \text{ in } J\}\text{-vol}(J)} \text{ for every } J \subset [0,1]^s$$

$$\mathbf{D_N^* = \sup_{E^*} |R_N(J)|},$$

E^* - the set of all rectangles with a vertex in zero.

- A s-dimensional sequence is called quasirandom if

$$\mathbf{D_N^* \leq c(\log N)^s N^{-1}}$$

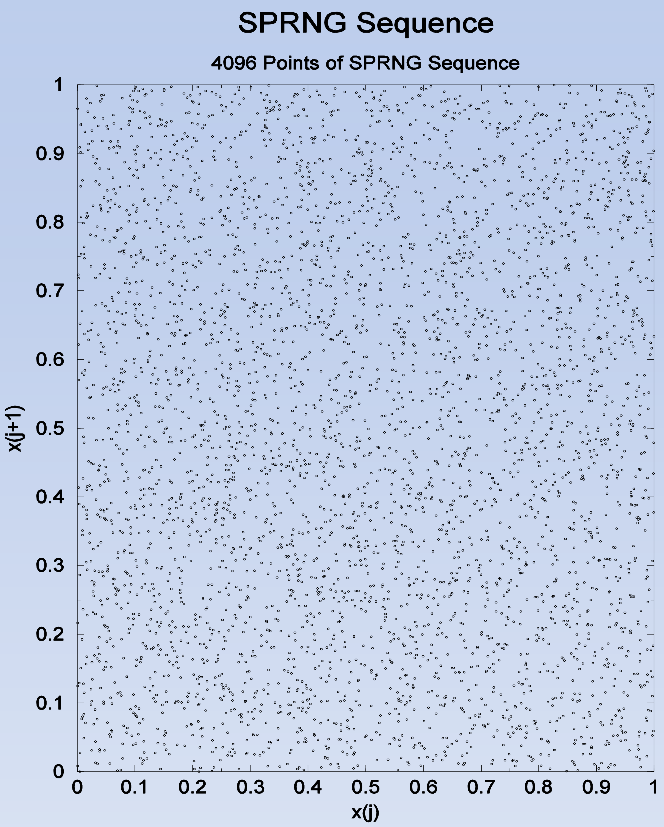
- Koksma-Hlawka inequality (for integration):

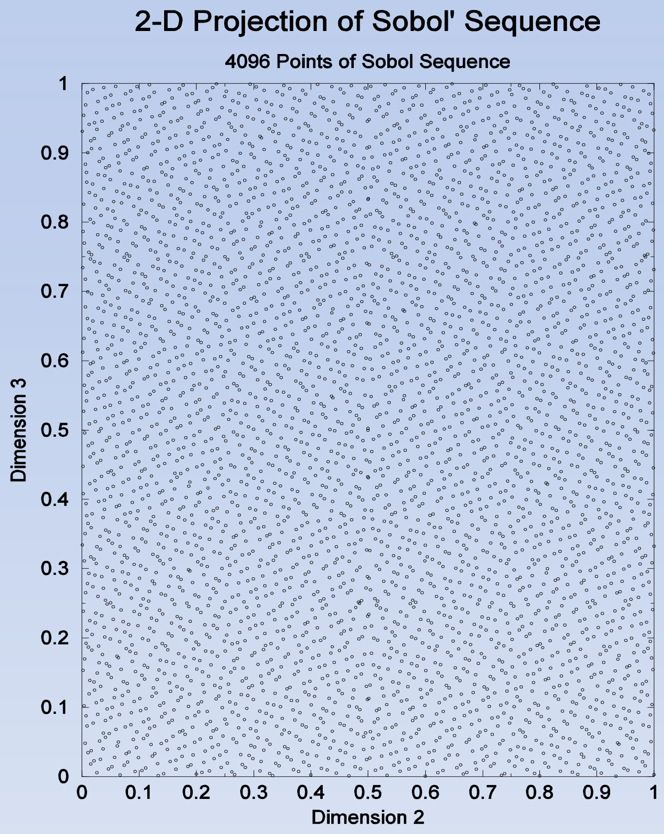
$$\mathbf{\varepsilon[f] \leq V[f]\, D_N^*}$$

(where V[f] is the variation in the sense of Hardy-Kraus)

- The order of the error is $O((\log N)^s N^{-1})$

PRNs and QRNs





Some facts

- Discrepancy of real random numbers:
  $D^*_N = O(N^{-1/2} (\log \log N)^{-1/2})$
- Klaus F. Roth (Fields medal 1958) proved the following lower bound for star discrepancy of N points in s dimensions:
  $D^*_N \geq O(N^{-1} (\log N)^{(s-1)/2})$
- Sequences (indefinite length) and point sets have different "best" discrepancies:
  - Sequence: $D^*_N \leq O(N^{-1} (\log N)^{s-1})$
  - Point set: $D^*_N \leq O(N^{-1} (\log N)^{s-2})$

Most often used sequences (Halton Sequence)

- Let n be an integer presented in base p. The p-ary radical inverse function is defined as

$$\phi_p(n) \equiv \frac{b_0}{p} + \frac{b_1}{p^2} + \ldots + \frac{b_m}{p^{m+1}}$$

where $\boldsymbol{p}$ is prime and $\boldsymbol{b_i}$ comes from:

$$n = b_0 + b_1 p + \ldots + b_m p^m, \quad 0 \le b_i < p$$

- An s-dimensional Halton sequence (1960) is defined as:

$$(\phi_{p_1}(n), \phi_{p_2}(n), \ldots, \phi_{p_s}(n))$$

with $\boldsymbol{p_1\, p_2 \ldots, p_s}$ being relatively prime, and usually the first $\boldsymbol{s}$ primes

Most often used sequences (Sobol)

- Sobol sequence (1967) $\{x_n = (x_n^{(1)}, x_n^{(2)}, \ldots, x_n^{(s)})\}$
- The j-th coordinate of the n-th point of s-dimensional ***Sobol sequence*** $x_n = (x_n^{(1)}, x_n^{(2)}, \ldots, x_n^{(s)})$ is generated through the recursion:

$$x_n^{(j)} = b_1 v_1^{(j)} \otimes b_2 v_2^{(j)} \otimes \ldots b_w v_w^{(j)}$$

where $v_i^{(j)}$ is i-direction number for dimension j, and $\otimes$ is bit-by-bit exclusive-or operation (b_i are the coefficients of representation of n in base 2)

- How to determine $v_i^{(j)}$: for each dimension a different primitive polynomial is chosen and its coefficients are used to define:

$$v_i^{(j)} = a_1^{(j)} v_{i-1}^{(j)} \otimes \ldots \otimes a_{dj-1}^{(j)} v_{i-dj+1}^{(j)} \otimes v_{i-dj}^{(j)} \otimes v_{i-dj}^{(j)}/2^{dj},\ i > dj$$

Most often used sequences (4)

- The nth point of the FAURE sequence (1981) is:
  $x_n = (\phi_b(P^0\mathbf{n}), \phi_b(P^1\mathbf{n}), . . . ,\phi_b(P^{n-1}\,\mathbf{n}))$,
  where b is a prime >= s and P^j are powers of Pascal matrix modulo b, and $\mathbf{n} = (n_0, n_1, . . . , n_m)^T$
- The complexity to generate one point of s-dimensional Faure sequence is $O(s(\log_b(n))^2)$.
- Other sequences: Niederreiter, lattice point sets, ergodic dynamics, etc

Quasirandom Sequences and their scrambling

- Unfortunately, the coordinates of the quasirandom sequence points in high dimensions show correlations. A possible solution to this problem is the ***scrambling***.
- The purpose of scrambling:
  - To improve 2-D projections and the quality of quasirandom sequences in general
  - To provide practical method to obtain error estimates for QMC
  - To provide simple and unified way to generate quasirandom numbers for parallel computing environments
  - To provide more choices of QRN sequences with better (often optimal) quality to be used in QMC applications

Scrambling techniques

- Scrambling was first proposed by Cranley and Patterson (1979) who took lattice points and randomized them by adding random shifts to the sequences. Later, ***Owen*** (1998, 2002, 2003) and ***Tezuka*** (2002) independently developed two powerful scrambling methods through permutations
- Although many other methods have been proposed, most of them are modified or simplified Owen or Tezuka schemes (*Braaten and Weller, Atanassov, Matousek, Chi and Mascagni, Warnock*, etc.)
- There are two basic scrambling methods:
  - Randomized shifting
  - Digital permutations

  (Permuting the order of points within the sequence)
- The problem with Owen scrambling is its computational complexity

Scrambling

- *Digital permutations*: Let $(x^{(1)}_n, x^{(2)}_n, . . . , x^{(s)}_n)$ be any quasirandom point in $[0, 1)^s$, and $(z^{(1)}_n, z^{(2)}_n, . . . , z^{(s)}_n)$ is its scrambled version. Suppose each $x^{(j)}_n$ has a b-ary representation $x^{(j)}_n, =0. x^{(j)}_{n1}\, x^{(j)}_{n2} \dots x^{(j)}_{nK}, \dots$ with K defining the number of digits to be scrambled. Then

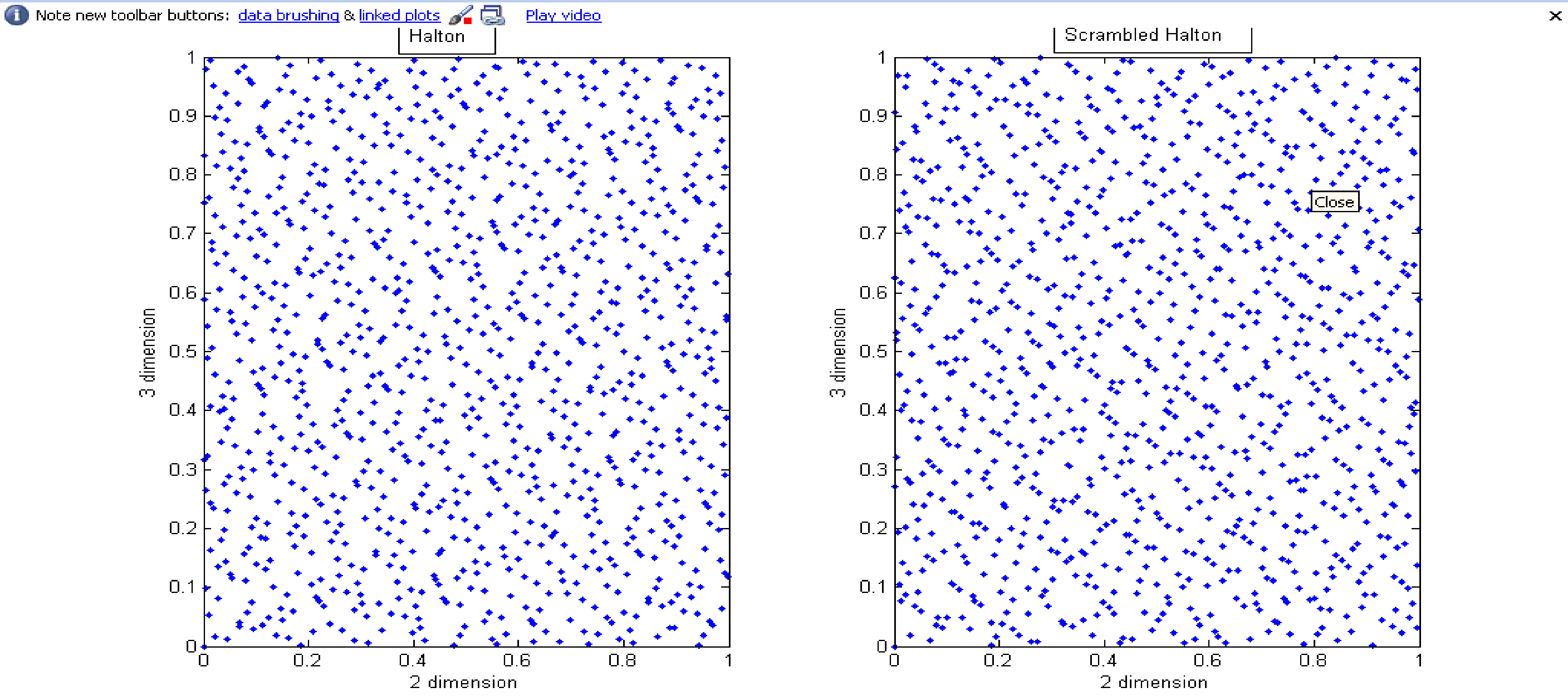
  $z^{(j)}_n = \sigma(x^{(j)}_n)$, where $\sigma=\{\Phi_1, \dots, \Phi_K\}$ и Φ_i, is a uniformly chosen permutation of the digits $\{0,1,\dots,b-1\}$.

- *Randomized shifting* has the form

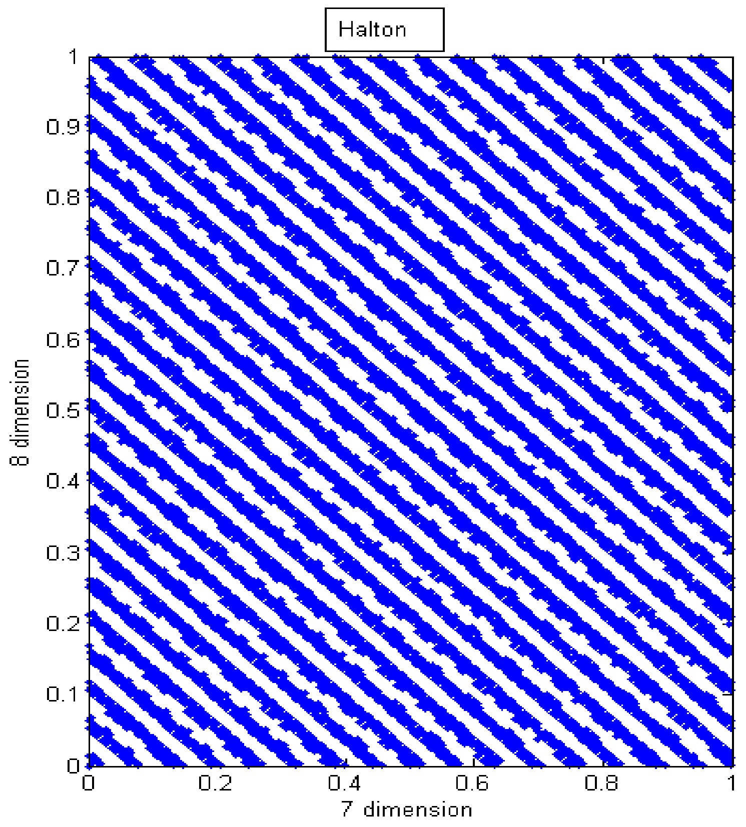
  $$z_n = x_n + r \pmod 1,$$

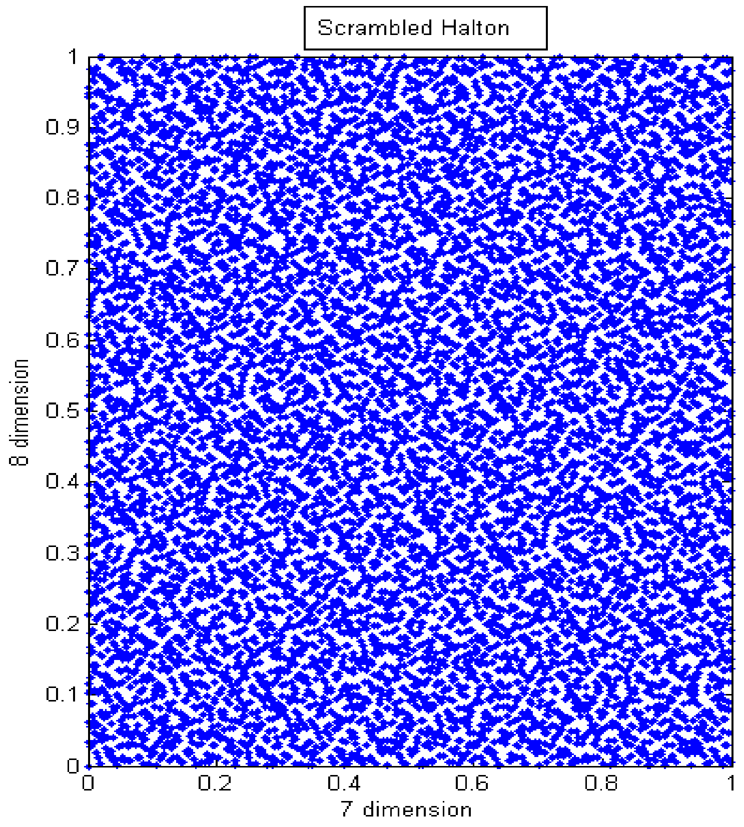
  where x_n is any quasirandom number in $[0, 1)^s$ and r is a single s-dimensional pseudorandom number.

Two-dimensional projection of Halton sequence and scrambled Halton sequence (dimension 3)

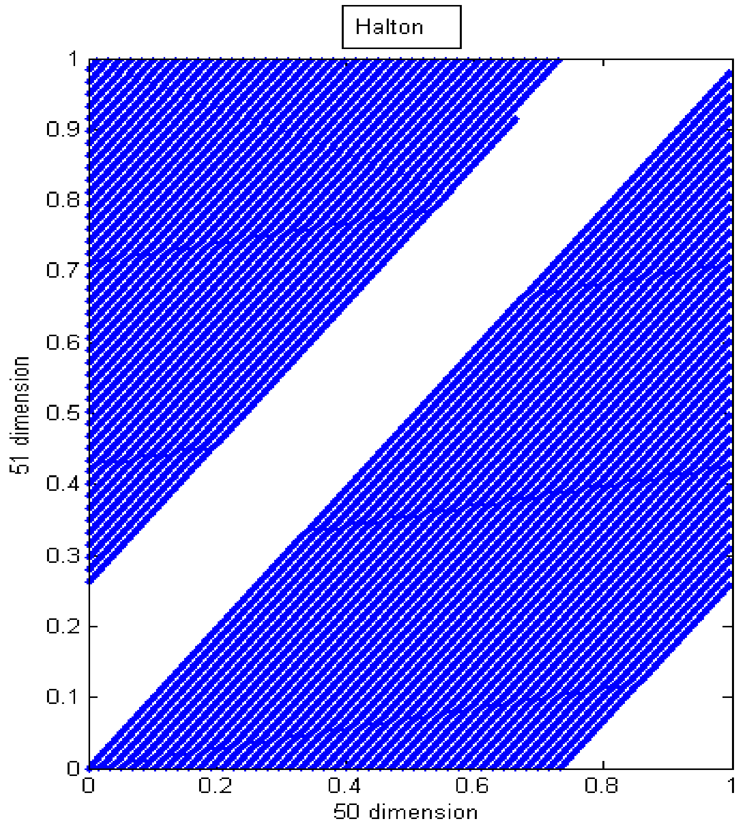


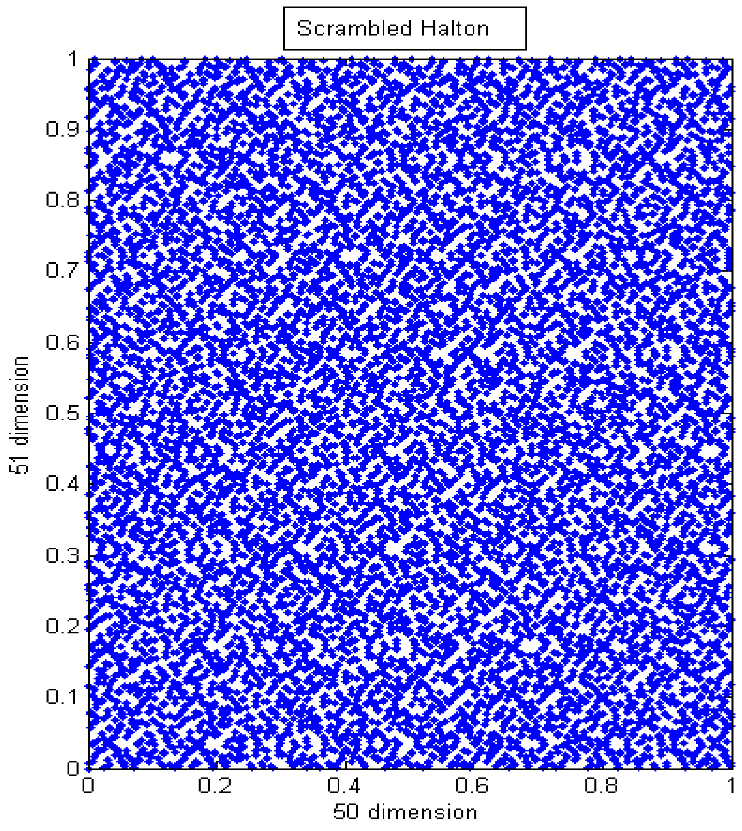
Two-dimensional projection of Halton sequence and scrambled Halton sequence (dimension 8)



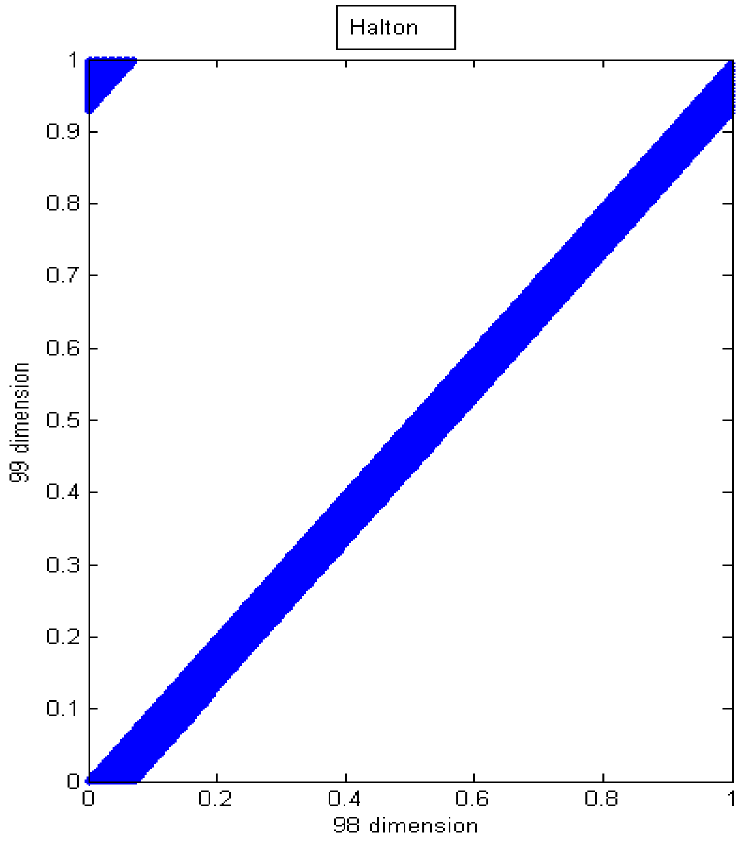


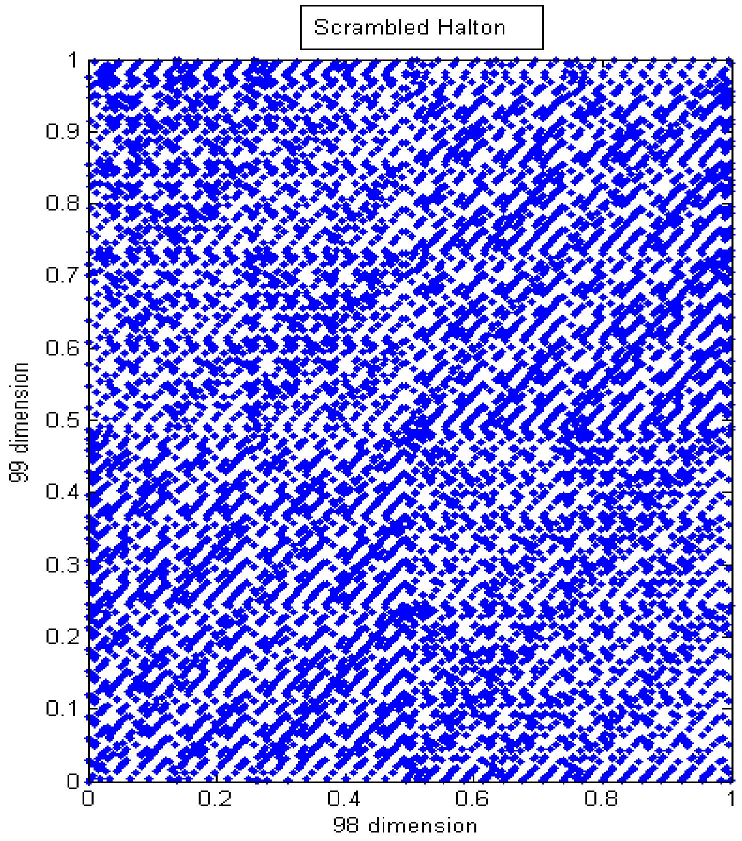
Two-dimensional projection of Halton sequence and scrambled Halton sequence (dimension 50)





Two-dimensional projection of Halton sequence and scrambled Halton sequence (dimension 99)





Simulation of electron transport in semiconductors (SET)

- SET solves various computationally intensive problems which describe ultrafast carrier transport in semiconductors
  - We consider the problem of a highly non-equilibrium electron distribution which propagates in a semiconductor or quantum wire
  - The electrons, which can be initially injected or optically generated in the wire, begin to interact with three-dimensional phonons
  - In the general case, a Wigner equation for nanometer and femtosecond transport regime is derived from a three equations set model based on the generalized Wigner function.
  - The complete Wigner equation poses serious numerical challenges. Two versions of the equation corresponding to simplified physical conditions are considered: the Wigner-Boltzmann equation and the homogeneous Levinson (or Barker-Ferry) equation.
- SET studies memory and quantum effects during the relaxation process due to electron-phonon interaction in semiconductors

SET: Quantum-kinetic equation (inhomogeneous case)

The integral form of the equation:

$$f_w(z,k_z,t) = f_{w,0}(z - \frac{\hbar k_z}{m}t, k_z) +$$
$$+ \int_0^t dt'' \int_{t''}^t dt' \int_G d^3\mathbf{k}' \{K_1(k_z,\mathbf{k}',t',t'') f_w(z + h(k_z,q'_z,t,t',t''), k'_z, t'')\}$$
$$+ \int_0^t dt'' \int_{t''}^t dt' \int_G d^3\mathbf{k}' \{K_2(k_z,\mathbf{k}',t',t'') f_w(z + h(k_z,q'_z,t,t',t''), k_z, t'')\}$$

$$h(k_z,q'_z,t,t',t'') = -\frac{\hbar k_z}{m}(t-t'') + \frac{\hbar q'_z}{2m}(t'-t'')$$

Kernels:

$$K_1(k_z,\mathbf{k}',t',t'') = S(k'_z,k_z,t',t'',\mathbf{q}'_\perp) = -K_2(\mathbf{k}',k_z,t',t'')$$

$$S(k'_z,k_z,t',t'',\mathbf{q}'_\perp) = \frac{2V}{(2\pi)^3}|G(\mathbf{q}'_\perp)\mathcal{F}(\mathbf{q}'_\perp,k_z-k'_z)|^2 \times$$
$$\left[(n(\mathbf{q}')+1)\cos\left(\frac{\epsilon(k_z)-\epsilon(k'_z)+\hbar\omega_{\mathbf{q}'}}{\hbar}(t'-t'')\right)\right.$$
$$\left. + n(\mathbf{q}')\cos\left(\frac{\epsilon(k_z)-\epsilon(k'_z)-\hbar\omega_{\mathbf{q}'}}{\hbar}(t'-t'')\right)\right]$$

SET: Quantum-kinetic equation (cont.)

Bose function: $n_{\mathbf{q}'} = 1/(\exp(\hbar\omega_{\mathbf{q}'}/\mathcal{K}T) - 1)$

The phonon energy (*ħω*) depends on : $\mathbf{q}' = \mathbf{q}'_\perp + q'_z = \mathbf{q}'_\perp + (k_z - k'_z)$

Electron energy: $\varepsilon(k_z) = (\hbar^2 k_z^2)/2m$

The electron-phonon coupling constant according to Fröhlich polar optical interaction:

$$\mathcal{F}(\mathbf{q}'_\perp, k_z - k'_z) = -\left[\frac{2\pi e^2 \omega_{\mathbf{q}'}}{\hbar V}\left(\frac{1}{\varepsilon_\infty} - \frac{1}{\varepsilon_s}\right)\frac{1}{(\mathbf{q}')^2}\right]^{\frac{1}{2}}$$

The Fourier transform of the square of the ground state wave function:

$$G(\mathbf{q}'_\perp) = \int d\mathbf{r}_\perp e^{i\mathbf{q}'_\perp \mathbf{r}_\perp} |\mathbf{\Psi}(\mathbf{r}_\perp)|^2$$

$$|G(\mathbf{q}'_\perp)|^2 = |G(q'_x)G(q'_y)|^2 =$$
$$\left(\frac{4\pi^2}{q'_x a\left((q'_x a)^2 - 4\pi^2\right)}\right)^2 4\sin^2(aq'_x/2)\left(\frac{4\pi^2}{q'_y a\left((q'_y a)^2 - 4\pi^2\right)}\right)^2 4\sin^2(aq'_y/2)$$

MCMs for Markov chain based problems

- Consider the following problem :

  $\mathbf{u = Ku + f}$

- The formal solution is the truncated Neumann series (for $||K||<1$):

  $\mathbf{u_{k+1} = f + Kf + \ldots + K^{k-1}f + K^k u_0}$

  with truncation error $u_k - u = K^k\ (u_0 - u)$.

- We are interested to compute the scalar product

  $\mathbf{J(u) = (h,u)}$, h – given vector

- MCM: Define r.v. θ such that $\mathbf{E[\theta] = J(u)}$:

  $\boldsymbol{\theta[h] = h(\xi_0)/\pi(\xi_0)\ \Sigma_{j=0}^{\infty}\ Q_j f(\xi_j),\ j=1,2,\ldots}$

  here $\boldsymbol{\xi_0, \xi_1, \ldots}$ is a Markov chain (random walk) in $\mathbf{G \in R^d}$ with initial density $\boldsymbol{\pi(x)}$ and transition density **p(x,y)**, which is equal to the normalized kernel of the integral operator.

- We have to estimate the mathematical expectation

Quasirandom walk

- Quasi-MCM error:

  $\delta(\zeta) = \lim_{N\to\infty}(\zeta(\omega_i) - \int_{\Omega} \zeta(\omega)d\mu(\omega))$

  where $\zeta(\omega_i)$ – the estimated variable is the analog of r.v. in MCM; ω_i – an element of the quasirandom walks space
- *Chelson's theorem for quasirandom walks* :

  $\delta_N(\zeta(Q')) \leq V(\zeta \circ \Gamma^{-1}).(D^*_N(Q))$

  where $Q = \{\gamma_i\}$ is a sequence of vectors in $[0,1)^{dT}$, $Q' = \{\omega_i\}$ is a sequence of quasirandom walks generated from Q by the mapping Γ;
  - There is a convergence
  - Impractical error as:

  $D^*_N = O((\log N)^{dT}/N)$, where d is the dimension of the original problem and T is the length of the chain

SET: The method

$$J_g(f) \equiv (g, f) = \int_0^{\mathcal{T}} \int_D g(z, k_z, t) f_w(z, k_z, t) dz dk_z dt$$

$$(z, k_z) \in D = (-Q_1, Q_1) \times (-Q_2, Q_2), \quad t \in (0, \mathcal{T})$$

$$(i) \quad g(z, k_z, t) = \delta(z - z_0)\delta(k_z - k_{z,0})\delta(t - t_0)$$

$$(ii) \quad g(z, k_z, t) = \frac{1}{2\pi}\delta(k_z - k_{z,0})\delta(t - t_0)$$

$$(iii) \quad g(z, k_z, t) = \frac{1}{2\pi}\delta(z - z_0)\delta(t - t_0)$$

Backward time evolution of the numerical trajectories

Wigner function:

$$f_w(z, k_z, t)$$

Energy (or momentum) distribution:

$$f(k_z, t) = \int \frac{dz}{2\pi} f_w(z, k_z, t)$$

Density distribution:

$$n(z, t) = \int \frac{dk_z}{2\pi} f_w(z, k_z, t)$$

SET: Quasirandom approach

- We adopted a hybrid approach, where evolution times are sampled using scrambled Sobol sequence or modified Halton sequence, and space parameters are modeled using pseudorandom sequences
- Scrambled modified Halton sequence [Atanassov 2003, 2014]:

  $x_n^{(i)} = \sum_{j=0}^{m} \text{imod}\,(a_j^{(i)} k_i^{j+1} + b_j^{(i)}, p_i)\; p_i^{-j-1}$

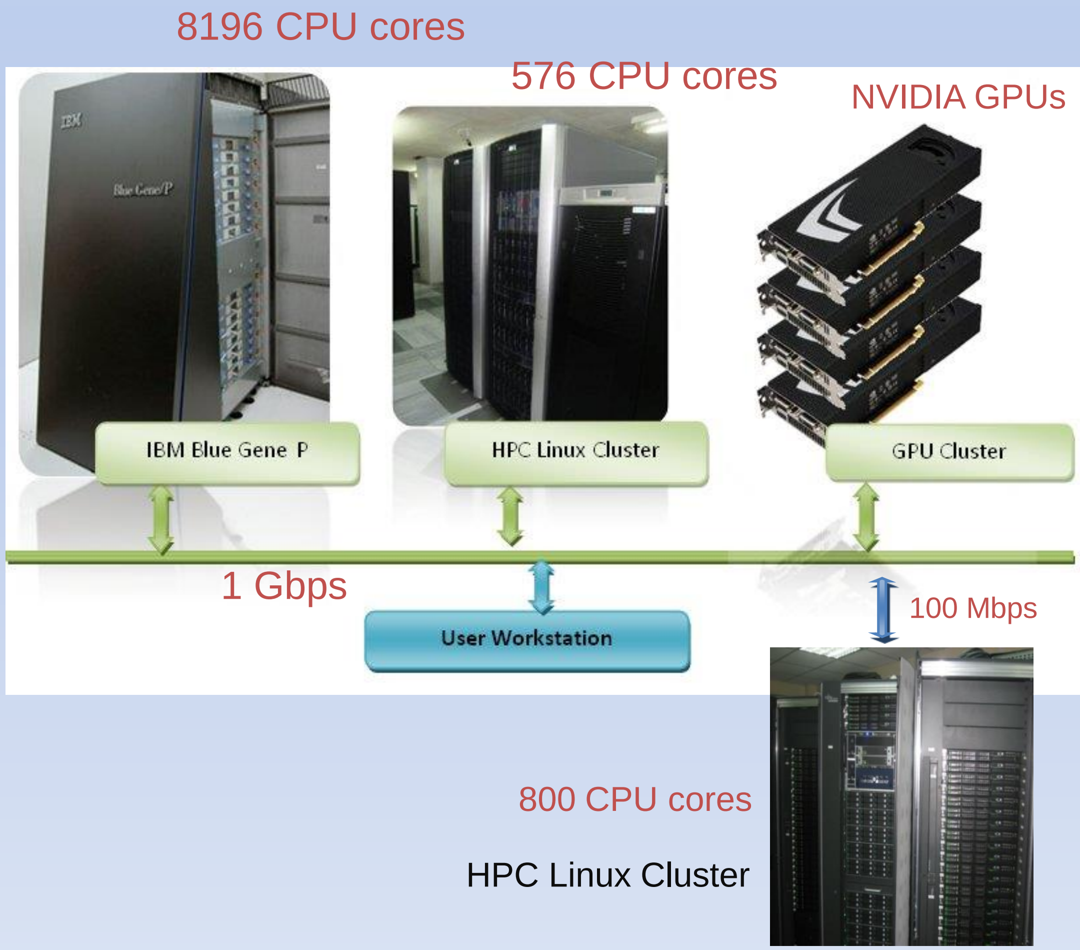
  (scramblers $b_j^{(i)}$, modifiers k_i in $[0, p_i - 1]$)
- The use of quasirandom numbers offers significant advantage because the rate of convergence is almost O(1/N) vs O(1/sqrt(N)) for regular pseudorandom numbers.
- The disadvantage is that it is not acceptable to lose some part of the computations and it therefore the execution mechanism should be more robust and lead to repeatable results.

Parallel implementation

- MC and QMC algorithms are perceived as computationally intensive, but naturally parallel. They can usually be implemented via the so-called *dynamic bag-of-work* model.
- In this model, a large task is split into smaller independent subtasks, which are then executed separately.
- One process or thread is designated as ``master'' and is responsible for the communications with the ``slave'' processes or threads, which perform the actual computations.
- The partial results are collected and used to assemble an accumulated result with smaller variance than that of a single copy.
- In our algorithm when the subtasks are of the same size, their computational time is also similar, i.e., we can also use static load balancing.
- Our parallel implementation uses MPI for the CPU-based parallelisation and CUDA for the GPU-based parallelisation

Target HPC Platforms

- The biggest HPC resource for research in Bulgaria is the supersupercomputer – **IBM BlueGene/P with 8192 cores**
- **HPC cluster with Intel CPUs and Infiniband** interconnection at IICT-BAS (vendors HP)
- In addition **GPU-enabled servers** equipped with state of the art GPUs are available for applications that can take advantage of them.



Bulgarian HPC Resources

- HPC Cluster at IICT-BAS
- 3 chassis HP Cluster Platform Express 7000, 36 blades BL 280c, dual Intel Xeon X5560 @ 2.8Ghz (total 576 cores), 24 GB RAM
- 8 servers HP DL 380 G6, dual Intel X5560 @ 2.8 GHz, 32 GB RAM
- Fully non-blocking DDR Infiniband interconnection
- Voltaire Grid director 2004 non-blocking DDR Infiniband switch,
- 2 disk arrays with 96 TB, 2 lustre fs
- Peak performance 3.2 TF, achieved performance more than 3TF, 92% efficiency.
- 2 HP ProLiant SL390s G7 Servers with 7 M2090 graphic cards

Scalability study (1)

- Our focus was to achieve the optimal output from the hardware platforms that were available to us. Achieving good scalability depends mostly on avoiding bottlenecks and *using good parallel pseudorandom number generators and generators for low-discrepancy sequences*. Because of the high requirements for computing time we took several actions in order to achieve the optimal output.
- The parallelization has been performed with MPI. Different version of MPI were tested and we found that the particular choice of MPI does not change much the scalability results. This was fortunate outcome as it allowed porting to the Blue Gene/P architecture without substantial changes.
- Once we ensured that the MPI parallelization model we implemented achieves good parallel efficiency, we concentrated on achieving the best possible results from using single CPU core.
- We performed profiling and benchmarking, also tested different generators and compared different pseudo-random number generators and low-discrepancy sequences.

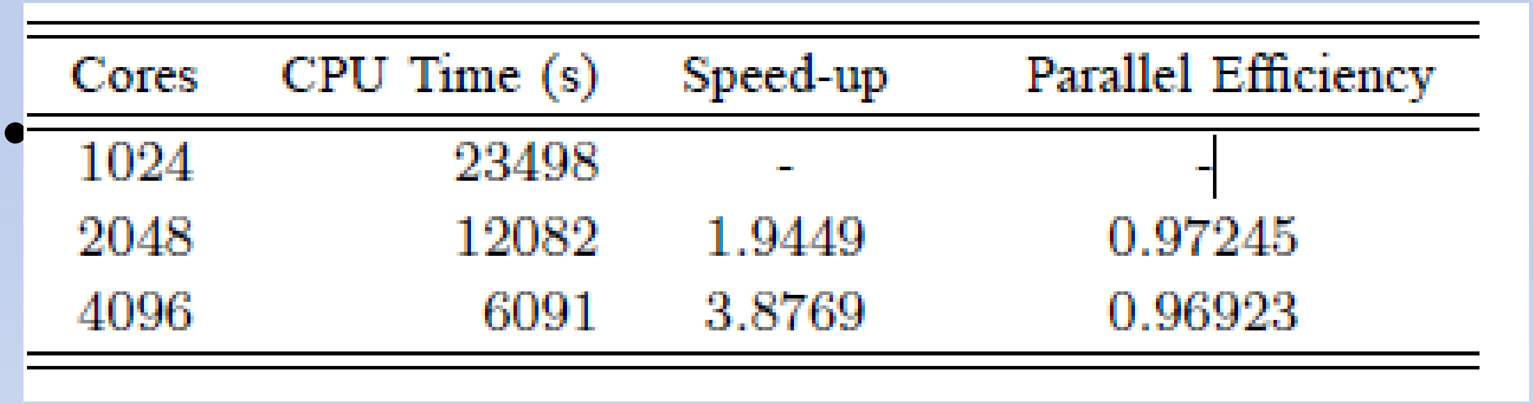
Scalability study (2)

- We tested various compilers and we concluded that the *Intel compiler currently provides the best results for the CPU version* running at our Intel Xeon cluster. For the IBM Blue Gene/P architecture the obvious choice was the *IBM XL compiler* suite since it has advantage versus the GNU Compiler Collection. For the GPU-based version that we developed recently we relay on the *C++ compiler supplied by NVIDIA.*
- For all the chosen compilers we performed tests to choose the best possible compiler and linker options. For the Intel-based cluster one important source of ideas for the options was the website of the SPEC tests, where one can see what options were used for each particular sub-test of the SPEC suite. From there we also took the idea to perform two-pass compilation, where the results from profiling on the first pass were fed to the second pass of the compilation to optimise further.

Scalability study (3)

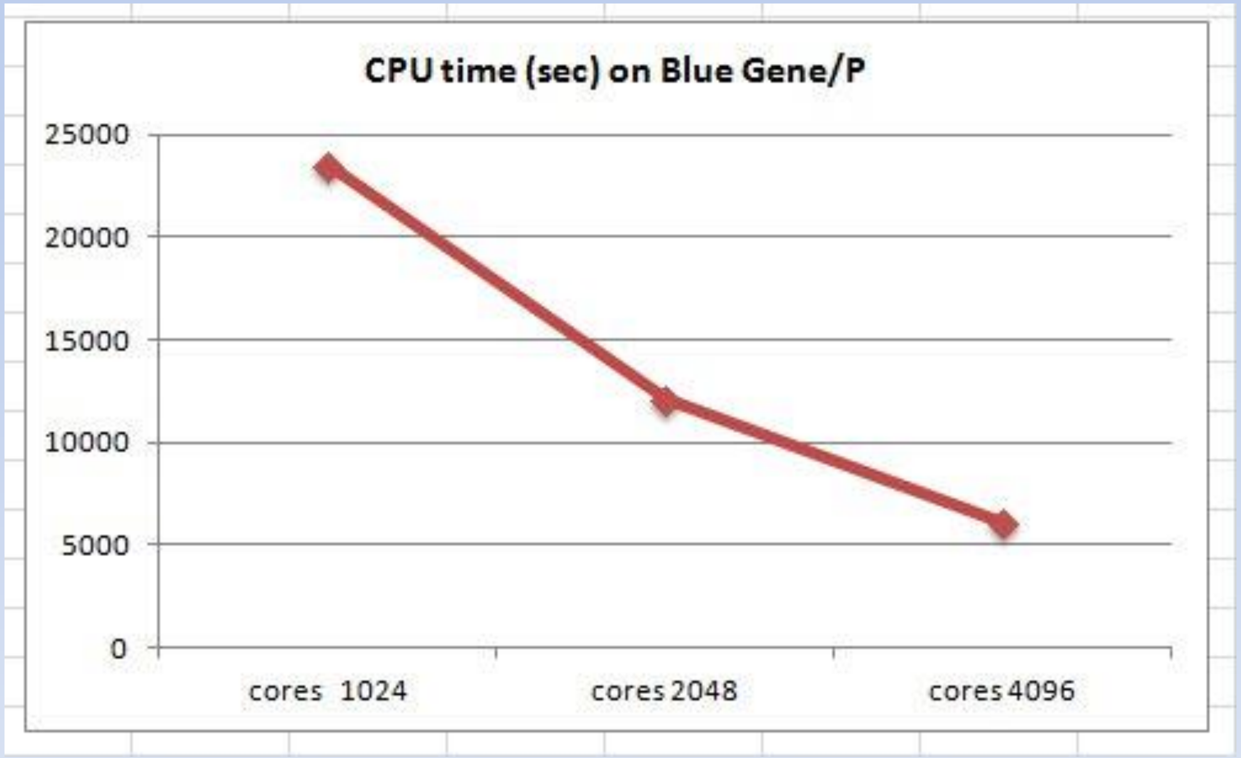
- For the HPCG cluster we also measured the performance of the parallel code ***with and without hyperthreading***. It is well known that hyperthreading does not always improve the overall speed of calculations, because the floating point units of the processor are shared between the threads and thus if the code is highly intensive in such computations, there is no gain to be made from hyperthreading. Our experience with other application yields such examples. But ***for the SET application we found about 30% improvement when hyperthreading is turned on***, which should be considered a good results and also shows that our overall code is efficient in the sense that most of it is now floating point computations, unlike some earlier version where the gain from hyperthreading was larger.

Numerical results (Blue Gene/P)



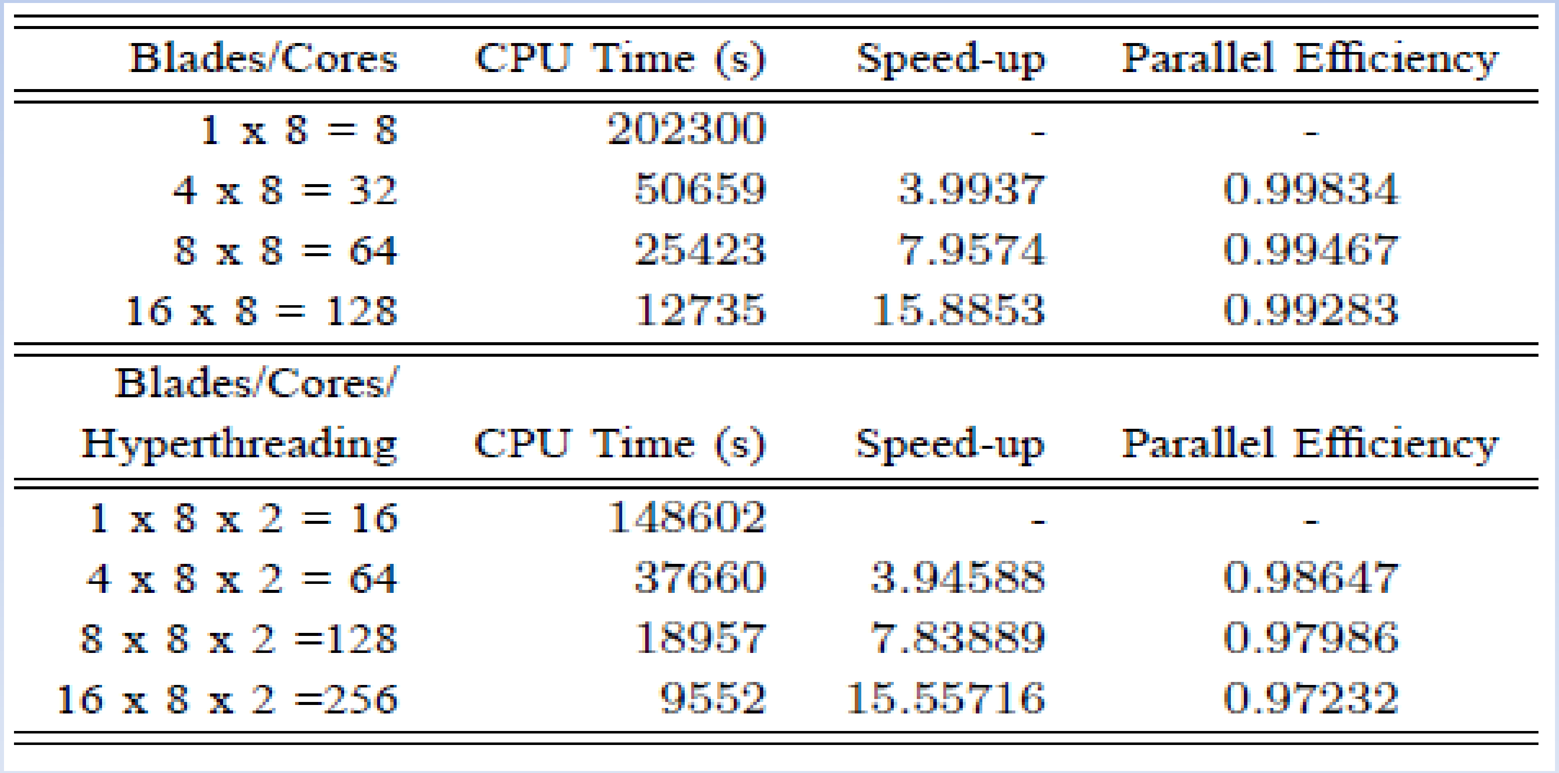
Cores	CPU Time (s)	Speed-up	Parallel Efficiency
1024	23498	-	-
2048	12082	1.9449	0.97245
4096	6091	3.8769	0.96923

Parallel efficiency (Blue Gene/P)



Numerical results (HPC cluster)

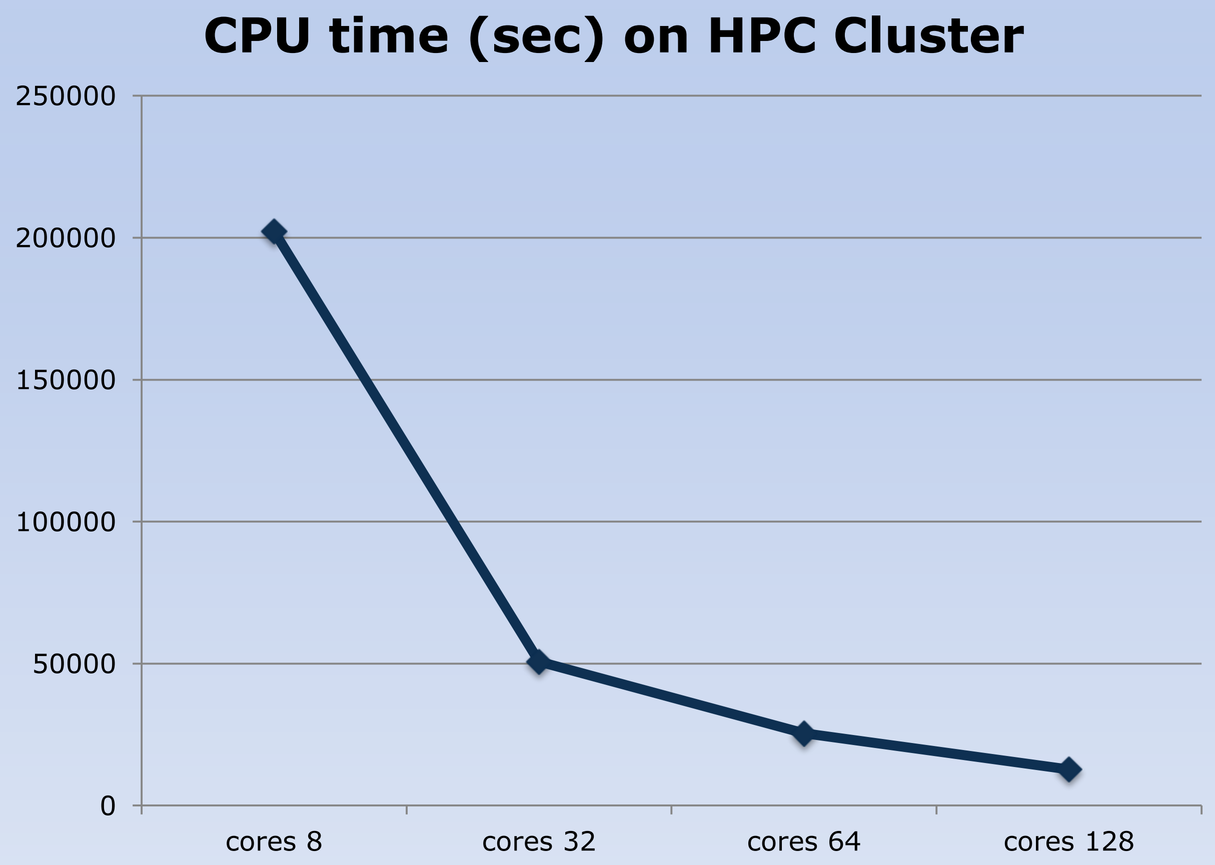
Results with electric field, 180fs, on Intel X5560 @2.8Ghz, Infiniband cluster



Blades/Cores	CPU Time (s)	Speed-up	Parallel Efficiency
1 x 8 = 8	202300	-	-
4 x 8 = 32	50659	3.9937	0.99834
8 x 8 = 64	25423	7.9574	0.99467
16 x 8 = 128	12735	15.8853	0.99283

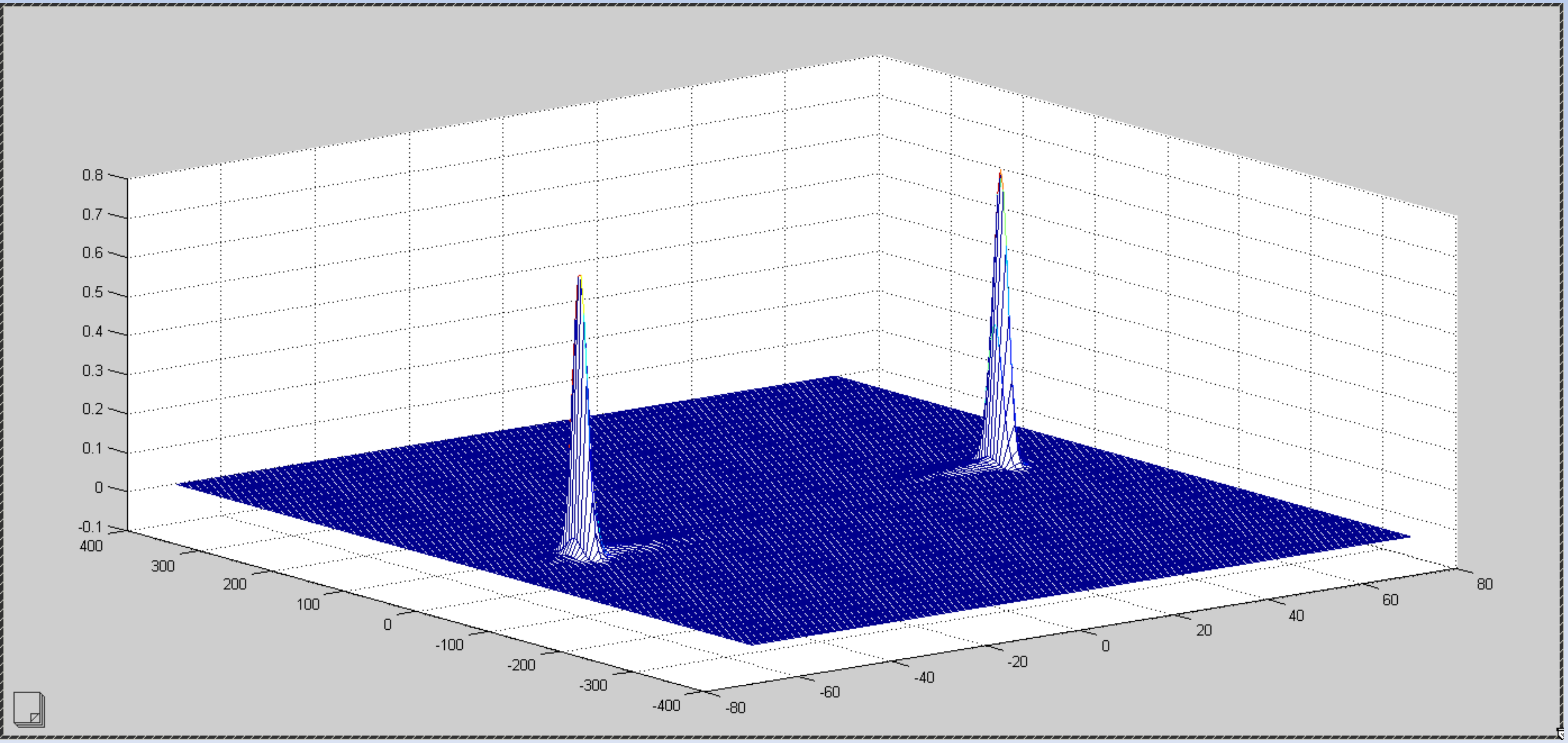
Blades/Cores/ Hyperthreading	CPU Time (s)	Speed-up	Parallel Efficiency
1 x 8 x 2 = 16	148602	-	-
4 x 8 x 2 = 64	37660	3.94588	0.98647
8 x 8 x 2 =128	18957	7.83889	0.97986
16 x 8 x 2 =256	9552	15.55716	0.97232

Parallel efficiency (HPC cluster)



Numerical results

Example results for the wigner function



Implementation using GPGPU

- Graphics cards have large number of cores. For NVIDIA cards one can use CUDA for parallel computations.
- Parallel processing on such cards is based upon splitting the computations between grid of threads.
- We use threadsize of 256, which is optimal taking into account relatively large number of registers.
- Generators for the scrambled Sobol sequence and modified Halton sequence have been developed and tested. For Monte Carlo we use CURAND

The GPGPU-based version

- Main target system: HP ProLiant SL390s G7
  - 2 Intel(R) Xeon(R) CPU   E5649 @ 2.53GHz
  - 96 GB RAM
  - Up to 8 NVIDIA Tesla (Fermi) cards, currently 6 M2090 cards
- Properties of the M2090 GPU device (Fermi):
  - 6 GB GDDR5 ECC RAM, 177 GB/s memory bandwidth
  - 512 GPU threads
  - 665 Gflops in double precision/1331 Gflops in single precision
- Our codes works on devices with support for double precision (devices with capabilities 1.3 and 2.0 used).

The GPGPU-based version

- Observations from running the GPGPU-based version:
- Threadsize of 256 seems optimal
- Significant number of divergent warps due to logical operators.
- Around 93 % parallel efficiency achieved when 6 cards were running computations for 10^8 samples in parallel.

Numerical results (GPU version)

Results with electric field, 180fs, same discretization as above:

- ✓ 67701 seconds for one M2090 card, 10^9 trajectories.
- ✓ One M2090 card is slightly slower than 4 Blades of the cluster without hyper-threading.
- ✓ 6 M2090 cards are faster than 16 blades of the cluster without hyperthreading and slightly slower than 16 blades with hyperthreading enabled.

Conclusions and future work

- The best results were achieved when using scrambled Sobol/Halton sequence for evolution times and PRNs for space parameters
- From our testing we concluded that hyperthreading should be used when available, two passes of compilation should be used for the Intel compiler targeting Intel CPUs, and that the application is scalable to the maximum number of available cores/threads at our disposable.
- Future work: study of energy aware performance using appropriate metrics.