Monte Carlo Methods for Electron Transport: Scalability Study

www.hp-see.eu

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

HP-SEE

High-Performance Computing Infrastructure for South East Europe's Research Communities

OUTLINE

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Introduction
- Problem formulation
- 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
- Benchmarking results of the GPGPU version
- Conclusions and future work

Introduction

HP-SEE

High-Performance Computing Infrastructure for South East Europe's Research Communities

- The problem of *stochastic modelling 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.

Simulation of electron transport in semiconductors (SET)

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- SET solves various computationally intensive problems which describe ultrafast carrier transport in semiconductors using Monte Carlo simulations
  - 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.
  - These equations are analyzed with various MCMs using spherical and cylindrical transformations to reduce the dimensions in the momentum space
- SET studies memory and quantum effects during the relaxation process due to electron-phonon interaction in semiconductors

SET: Quantum-kinetic equation (inhomogeneous case)

HP-SEE
High-Performance Computing Infrastructure
arch Communities

The integral form of the equation:

$$f_w(z,k_z,t) = f_{w,0}(z - \frac{\hbar k_z}{m}t, k_z) +$$

$$+ \int_0^t dt'' \int_{t''}^t dt' \int_G d^3\mathbf{k}' \{K_1(k_z,\mathbf{k}',t',t'') f_w\left(z + h(k_z,q_z',t,t',t''), k_z', t''\right)\}$$

$$+ \int_0^t dt'' \int_{t''}^t dt' \int_G d^3\mathbf{k}' \{K_2(k_z,\mathbf{k}',t',t'') f_w\left(z + h(k_z,q_z',t,t',t''), k_z, t''\right)\}$$

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

Kernels:

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

$$S(k_z',k_z,t',t'',\mathbf{q}_\perp') = \frac{2V}{(2\pi)^3}|G(\mathbf{q}_\perp')\mathcal{F}(\mathbf{q}_\perp', k_z - k_z')|^2 \times$$

$$\Big[(n(\mathbf{q}')+1)\cos\left(\frac{\epsilon(k_z) - \epsilon(k_z') + \hbar\omega_{\mathbf{q}'}}{\hbar}(t'-t'')\right)$$

$$+ n(\mathbf{q}')\cos\left(\frac{\epsilon(k_z) - \epsilon(k_z') - \hbar\omega_{\mathbf{q}'}}{\hbar}(t'-t'')\right)\Big]$$

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:

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

$$|G(\mathbf{q}'_{\perp})|^2 = |G(q'_x)G(q'_y)|^2 =$$

$$\left(\frac{4\pi^2}{q'_x a\left((q'_x a)^2 - 4\pi^2\right)}\right)^2 4\sin^2(aq'_x/2)\left(\frac{4\pi^2}{q'_y a\left((q'_y a)^2 - 4\pi^2\right)}\right)^2 4\sin^2(aq'_y/2)$$

MCMs for Markov chain based problems

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- 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 $\mathbf{E[\theta] = J(u)}$:

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

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

- We have to estimate the mathematical expectation

MCM accuracy

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- 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

Quasirandom walk

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- 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* :

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

  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

Quasirandom sequences

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- The quasirandom sequences are deterministic sequences constructed to be as uniform 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 sequence is called quasirandom if

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

- Koksma-Hlawka inequality (for integration):

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

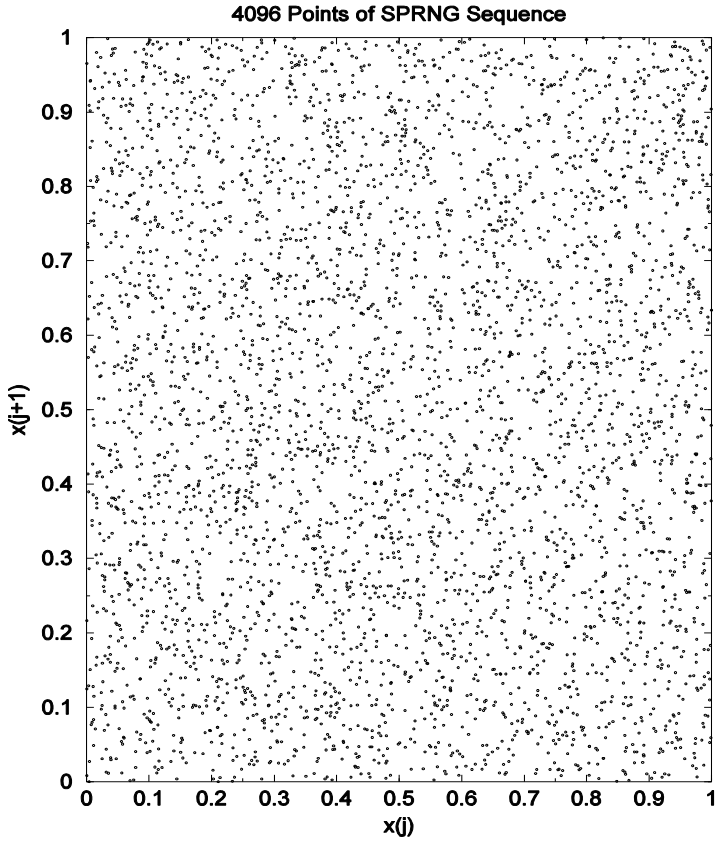
(where V[f] is the variation in the sense of Hardy-Kraus)

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

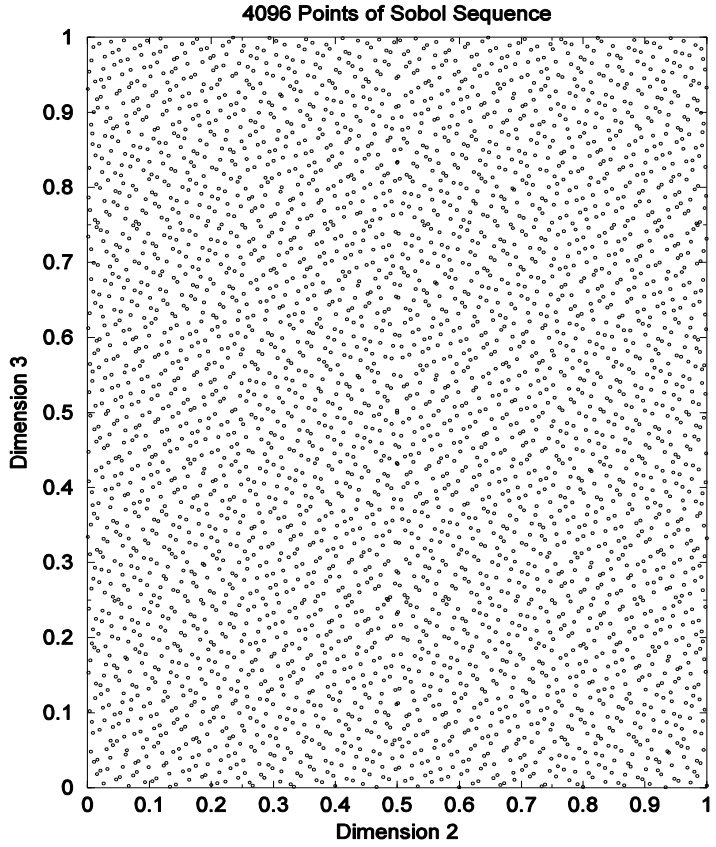
PRNs and QRNs

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

SPRNG Sequence



2-D Projection of Sobol' Sequence



Quasirandom Sequences and their scrambling

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Star discrepancy:
  - Quasirandom sequences: $\mathbf{D_N^* < c\,(\log N)^s\,N^{-1}}$
  - Random numbers: $\mathbf{D_N^* = O\,((\log\log N)^{-1/2}\,N^{-1/2})}$
- A few quasirandom sequences are currently widely used: Halton, Faure, Niederreiter and Soboĺ
- Unfortunately, the coordinates of the points in high dimensions show correlations. A possible solution to this problem is the so-called 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, distributed and grid-based computing environments
  - To provide more choices of QRN sequences with better (often optimal) quality to be used in QMC applications

Scrambling techniques

HP-SEE

High-Performance Computing Infrastructure for South East European Research Communities

- 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 three 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

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

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

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

The Halton Sequence

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- 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 p is prime and b_i comes from

$$n = b_0 + b_1 p + \ldots + b_m p^m, \qquad \text{with } 0 \le b_i < p$$

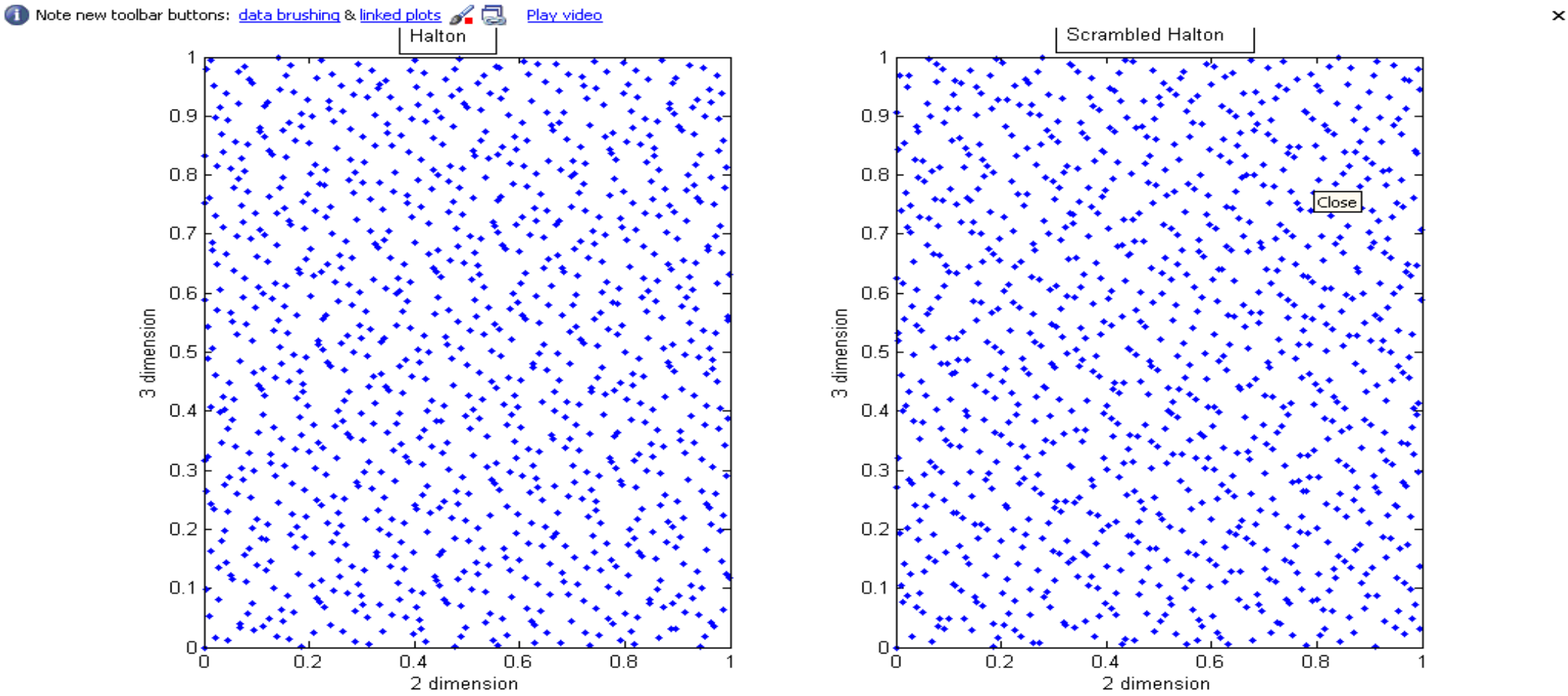
- An s-dimensional Halton sequence is defined as:

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

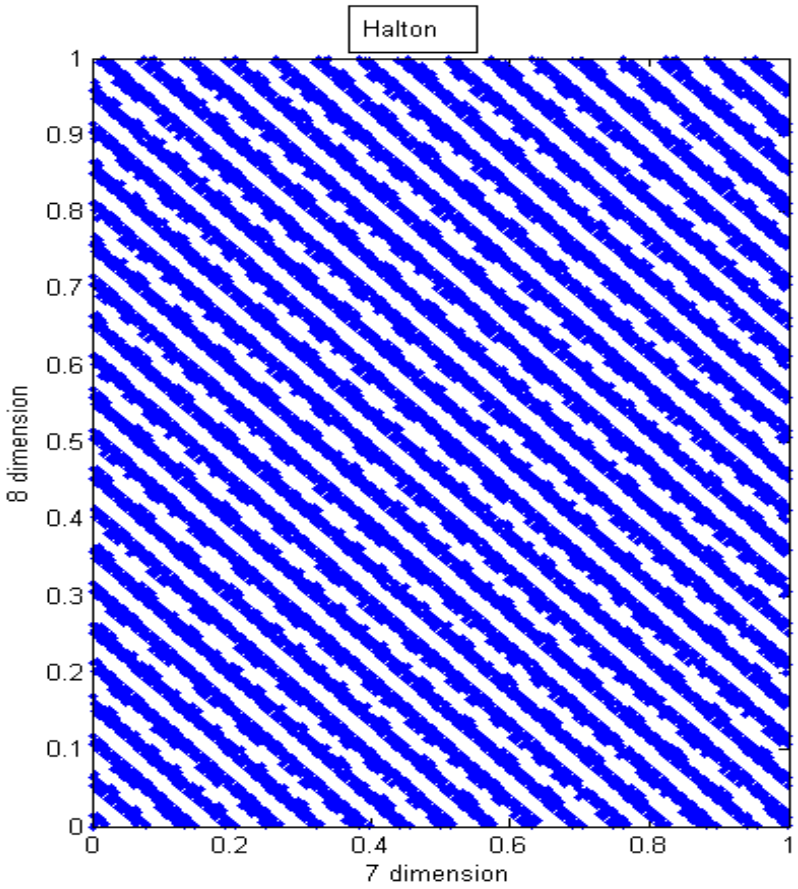
with $p_1\, p_2\, \ldots, p_s$ being relatively prime, and usually the first s primes

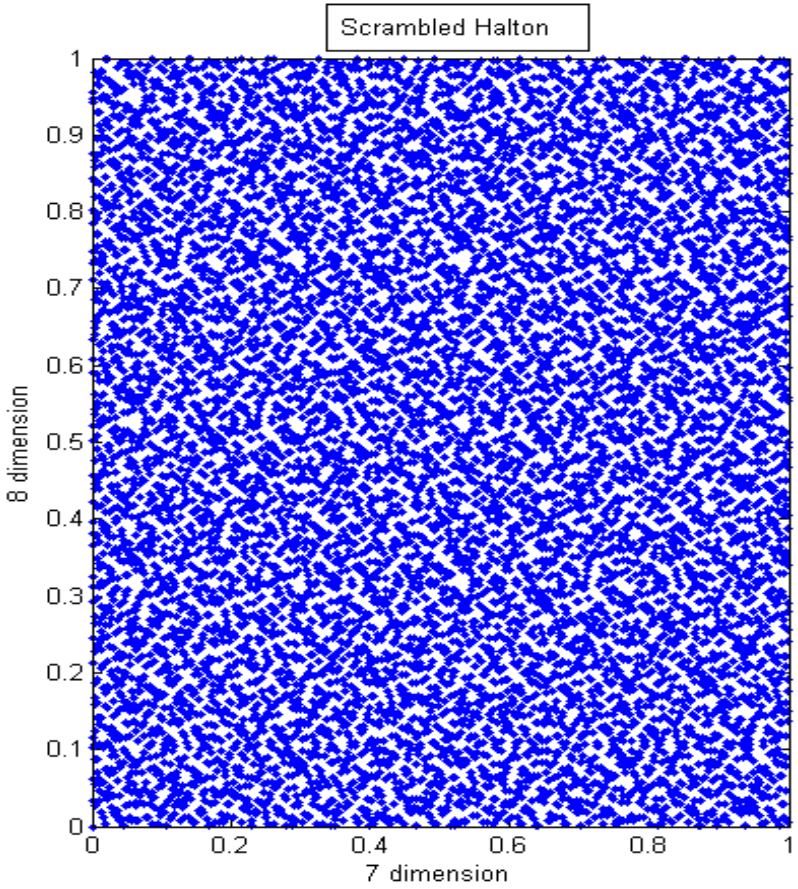
Two-dimensional projection of Halton sequence and scrambled Halton sequence (dimension 3)

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

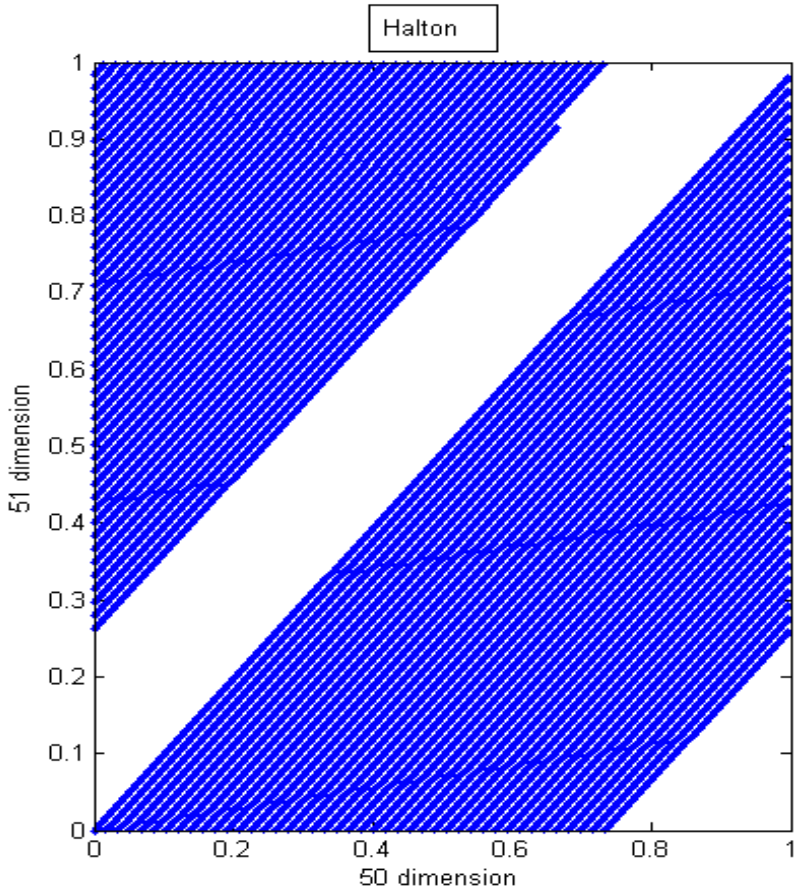


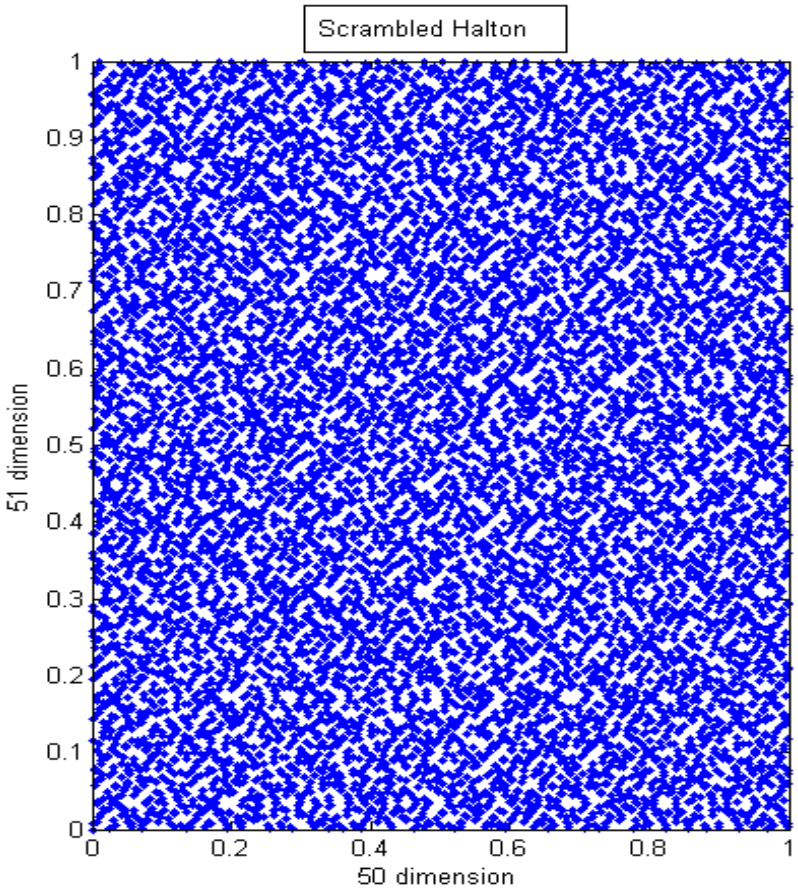
Two-dimensional projection of Halton sequence and scrambled Halton sequence (dimension 8)



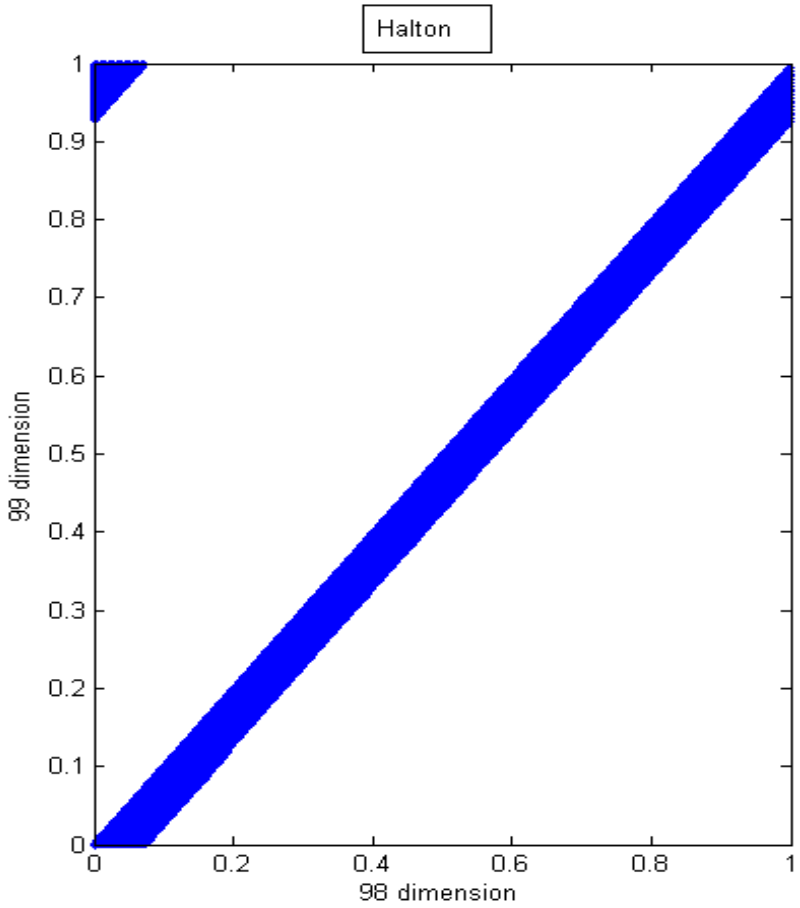


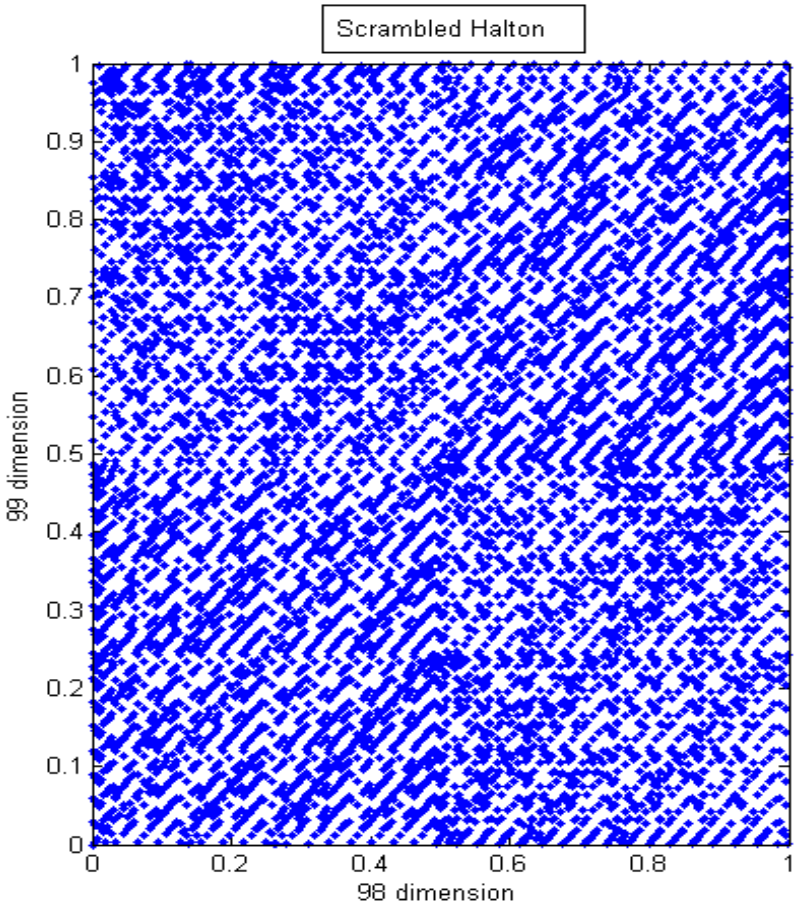
Two-dimensional projection of Halton sequence and scrambled Halton sequence (dimension 50)





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





Halton Sequence Correlations

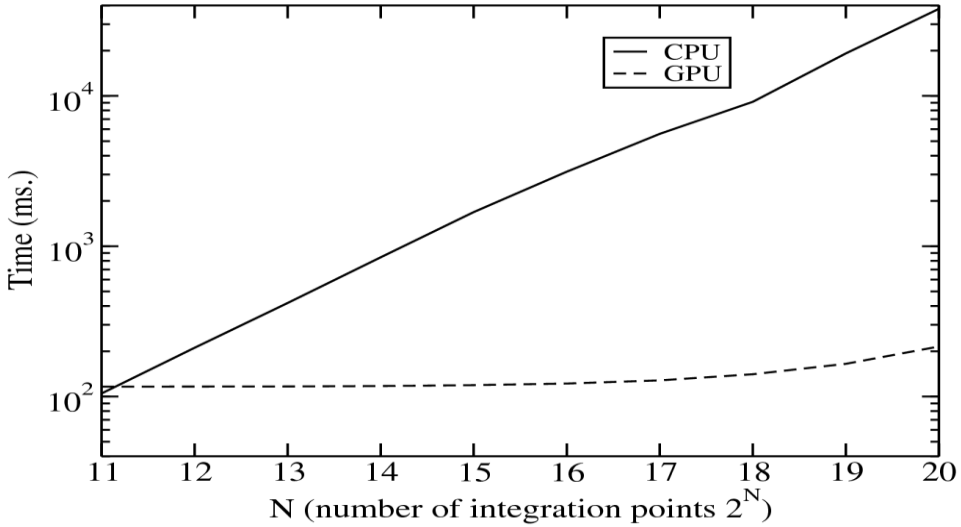
HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- The correlation between the radical inverse function with different, but close, bases corresponding to different dimensions causes the Halton sequence to have bad 2-D projections in those dimensions

- We want to quantitatively calculate these correlations so that we can find a method to improve this situation for the Halton sequence

Full scrambling

- Owen type of scrambling preserves the star discrepancy but is very time consuming
- We have developed GPU-based algorithms for Owen type of scrambling for Sobol sequence (2010, Atanassov, Karaivanova, Ivanovska)
- With this algorithm we achieved a reasonable time to produce the scrambled sequences



Linear Permutations

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Let $A = \phi_p(n) \equiv \frac{b_0}{p} + \frac{b_1}{p^2} + \ldots + \frac{b_m}{p^{m+1}}$, then a version of linear permutation applied to $\boldsymbol{A}$ is:

$$X = \frac{\pi_p(b_0)}{p} + \frac{\pi_p(b_1)}{p^2} + \ldots + \frac{\pi_p(b_m)}{p^{m+1}}$$

where $\boldsymbol{\pi_p(b_i) = (ab_i + g) \bmod p}$ and $\boldsymbol{0 < a \leq p\text{-}1,\ 0 \leq g \leq p\text{-}1}$

Since the $\boldsymbol{p}$ is different for each dimension the permutation $\boldsymbol{\pi_p(b_i)}$ will be different for each dimension

- We found an optimal value of $\boldsymbol{a}$ with $\mathbf{g}=0$ for different prime bases up to dimension 40
- We randomly choose nonzero *g*'s with the set of optimal *a*'s
  - This gives another randomization
  - The linear scrambling quality is guided by the *a*
- This set was used for error estimation giving smaller confidence intervals
- By using this optimal set, we can get 10%-30% improvement in error estimation

SET: Monte Carlo method

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

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

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

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

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

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

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: Monte Carlo Method (cont.)

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

Biased

MC estimator:

Weights:

$$(k_{zj}, t'_j, t_j) \in (-Q_2, Q_2) \times (t_j, t_{j-1}) \times (0, t_{j-1})$$

The Markov chain:

$$(k_{z0}, t_0) \to (k_{z1}, t'_1, t_1) \to \ldots \to (k_{zj}, t'_j, t_j) \to \ldots \to (k_{zs},, t'_s, t_s),\ j = 1, 2, \ldots, s$$

$$(z, k_{z0}, t_0): \quad p_{in}(z, k_z, t) = g(z,, k_z, t)$$

Initial density function

$$p_{tr}(\mathbf{k}, \mathbf{k}', t', t'') = p(\mathbf{k}'/\mathbf{k}) p(t, t', t'')$$

Transition density function:

SET: Monte Carlo method

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- The variance increases exponentially with respect to the relaxation time T.
- Achieving accurate results requires accumulating the results of billions of trajectories
- Improvements in variance and execution time can be achieved with low-discrepancy sequences (quasirandom numbers).
- The use of quasirandom numbers requires a robust and flexible implementation, since it is not feasible to ignore failures and missing results of some trajectories, unlike in Monte Carlo.
- GPU resources are efficient in computations using the low-discrepancy sequences of Sobol, Halton, etc.
- Variance reduction in case of pure MC can be achieved using different transition density functions.

SET: Quasirandom approach

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- 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]:

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

  (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.

SET: Monte Carlo modelling of semiconductor devices

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Variance reduction approach – because of the high variance, it is justified to study and optimize the transfer functions.
- Thus a parallel version of the genetic optimisation library *galib* was developed and succesfully run on the BlueGene/P.
- It was used to optimise the transfer function related to the evolution time (instead of constant).
- So far gains are not more than 20% but we are considering the possibility to optimise the other kernels, which are more complex and probably will lead to better results.

Parallel implementation

HP-SEE

High-Performance Computing Infrastructure for South East Europe's Research Communities

- The stochastic error for the Levinson or Barker-Ferry models has order $O(e^{c2t} N^{-1/2})$, where t is the evolution time. The stochastic error of the MC estimator has order $O(e^{c3t\hat{}2} N^{-1/2})$.
- Consequently lots of CPU power is needed when evolution time is above 100 femtoseconds.
- MC 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 MC 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

Parallel implementation of RNGs

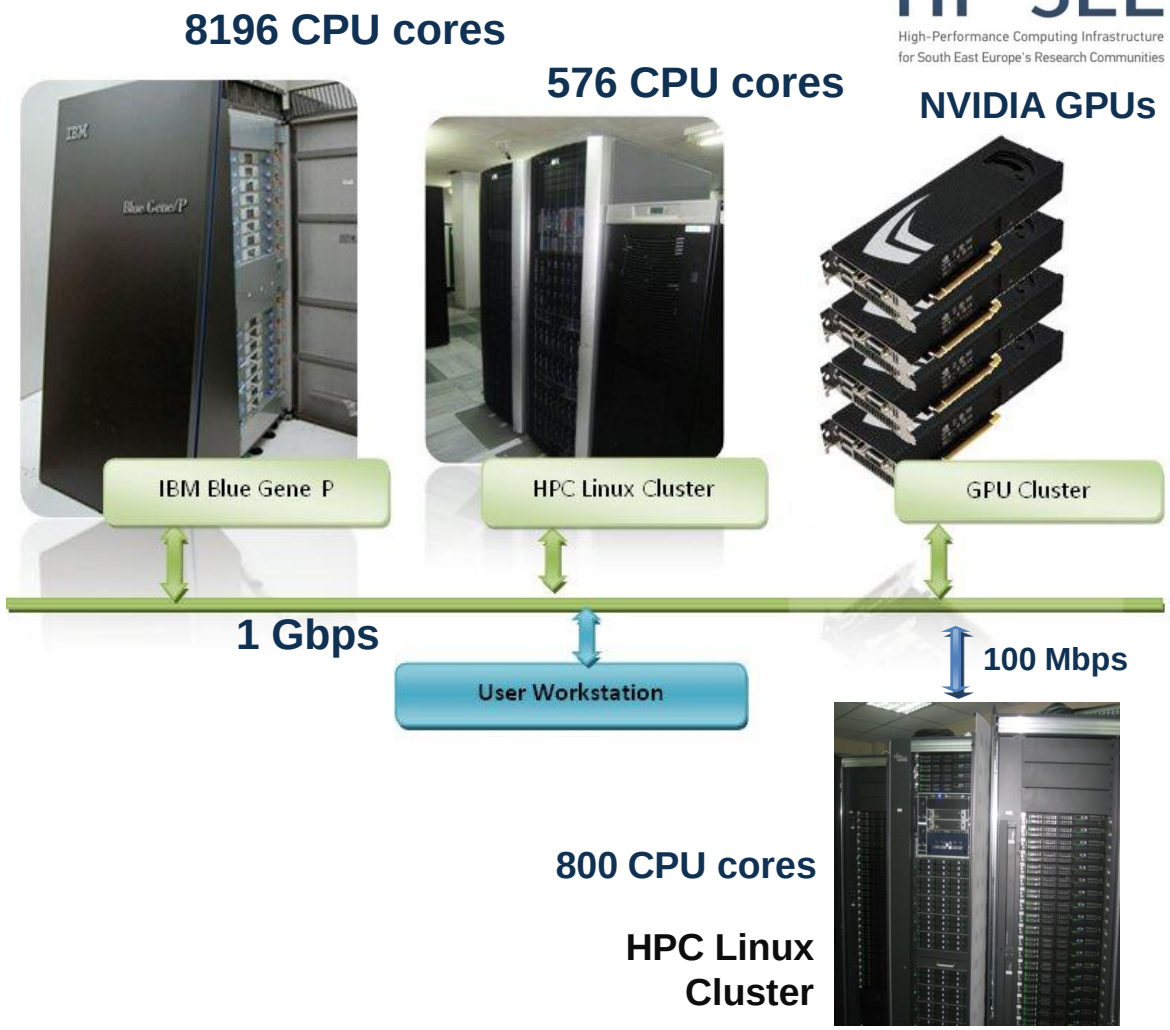
HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

Block with length L is assigned to each processor.

1. Blocking:
   - First block: $\{x_0, x_1, \dots, x_{L-1}\}$
   - Second block: $\{x_L, x_{L+1}, \dots, x_{2L-1}\}$
   - i-th block: $\{x_{(i-1)L}, x_{(i-1)L+1}, \dots, x_{iL-1}\}$
2. The leap-frog technique: define the leap ahead of $l = [Per(x_i)/L\ ]$:
   - First block: $\{x_0, x_l, x_{2l}, \dots, x_{(L-1)l}\}$
   - Second block: $\{x_1, x_{1+l}, x_{1+2l}, \dots, x_{1+(L-1)l}\}$
   - i-th block: $\{x_i, x_{i+l}, x_{i+2l}, \dots, x_{i+(L-1)l}\}$
3. Using distinct parameterized streams in parallel

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.



BG Blue Gene/P in Sofia

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- IBM Blue Gene/P –two racks, 2048 *PowerPC 450 processors (32 bits, 850 MHz), a total of 8192 cores;*
- A total of 4 TB random access memory;
- **16 I/O nodes** currently connected via fiber optics to a **10 Gb/s** Ethernet switch;
- Theoretical peak performance: Rpeak= **27.85 Tflops;**
- **Energy efficiency: 371.67 MFlops/W**
- 1 Gb/s Ethernet fiber optics link to Bulgarian NREN's Point-of-Presence at the IICT-BAS
- Operating System for front-end node: SUSE Linux Enterprise Server 10 (SLES 10), Service Pack 1 (BG/P)
- The Compute Nodes run OS Compute Node Kernel (CNK)
- 2 file servers, 12 TB storage

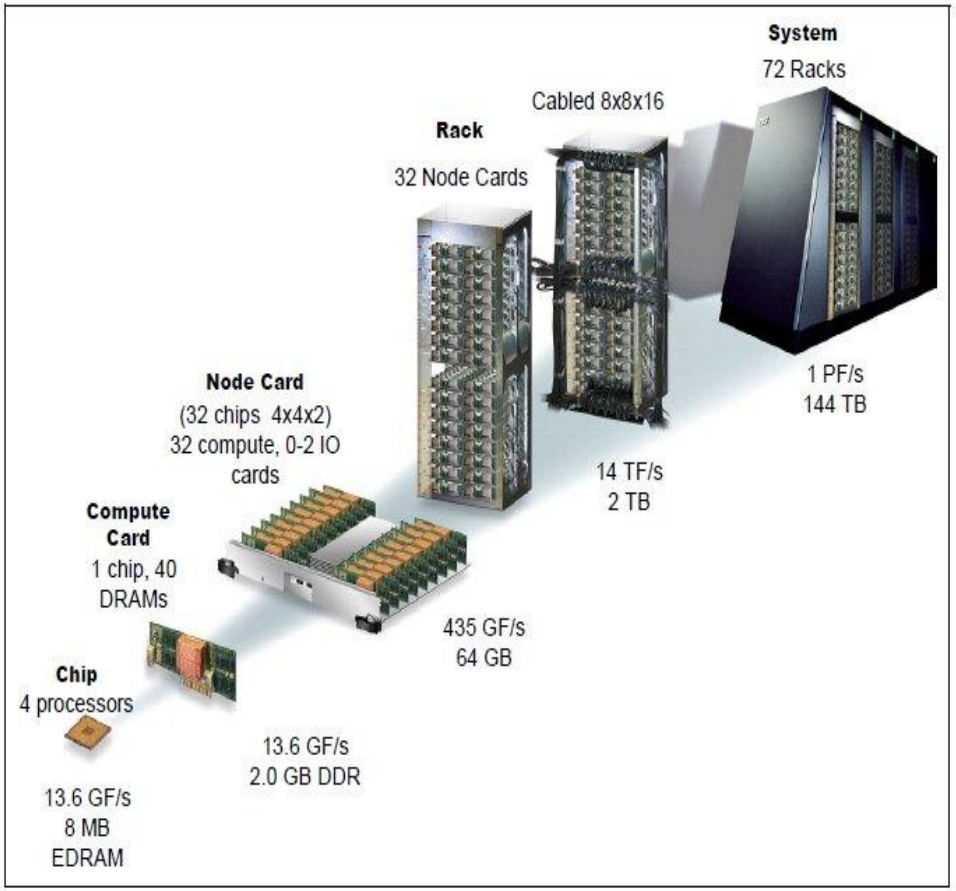


Figure 1-2 Blue Gene/P packaging

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)

HP-SEE

High Performance Computing Infrastructure for South East Europe's Research Communities

- 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)

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- 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 choosen 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)

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- 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 of the HP-SEE project 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.

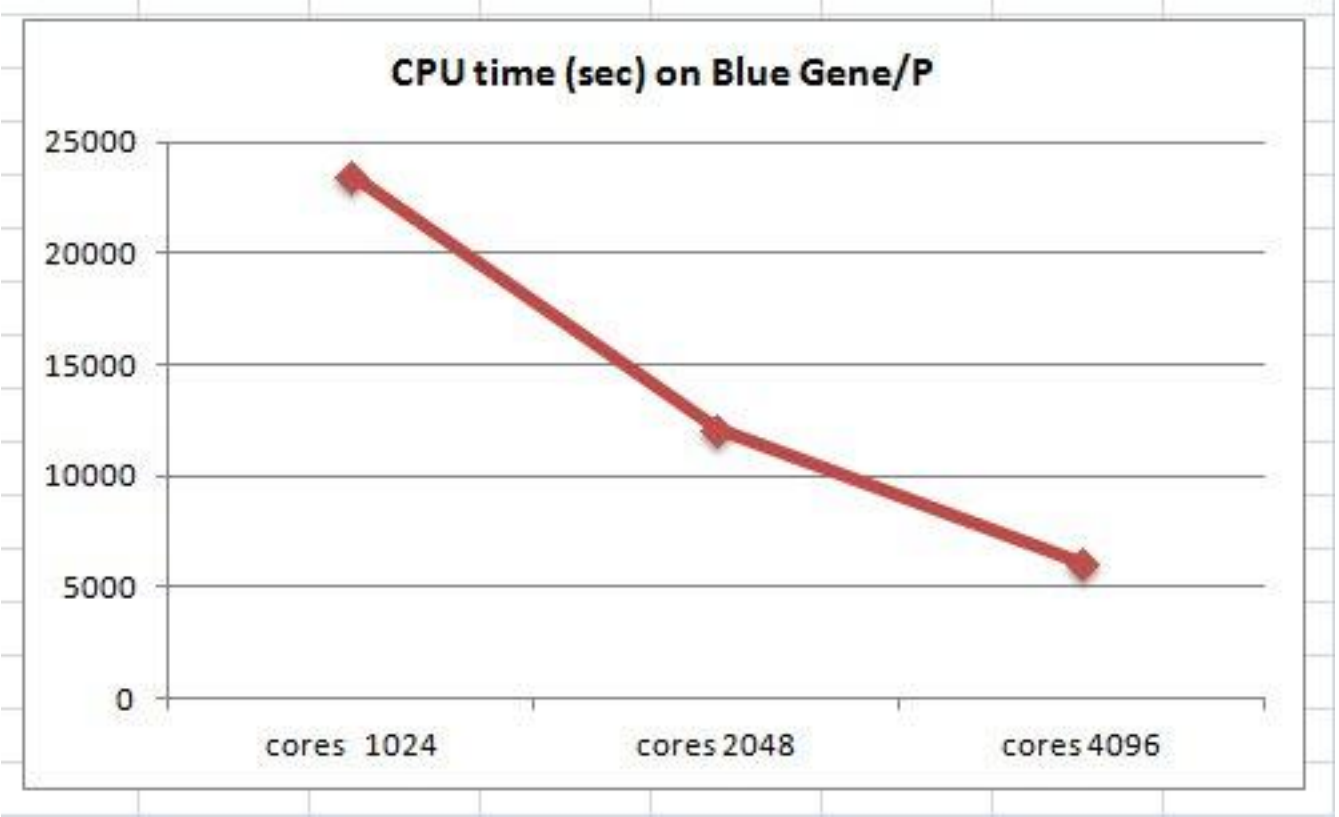
Numerical results

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Results on 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

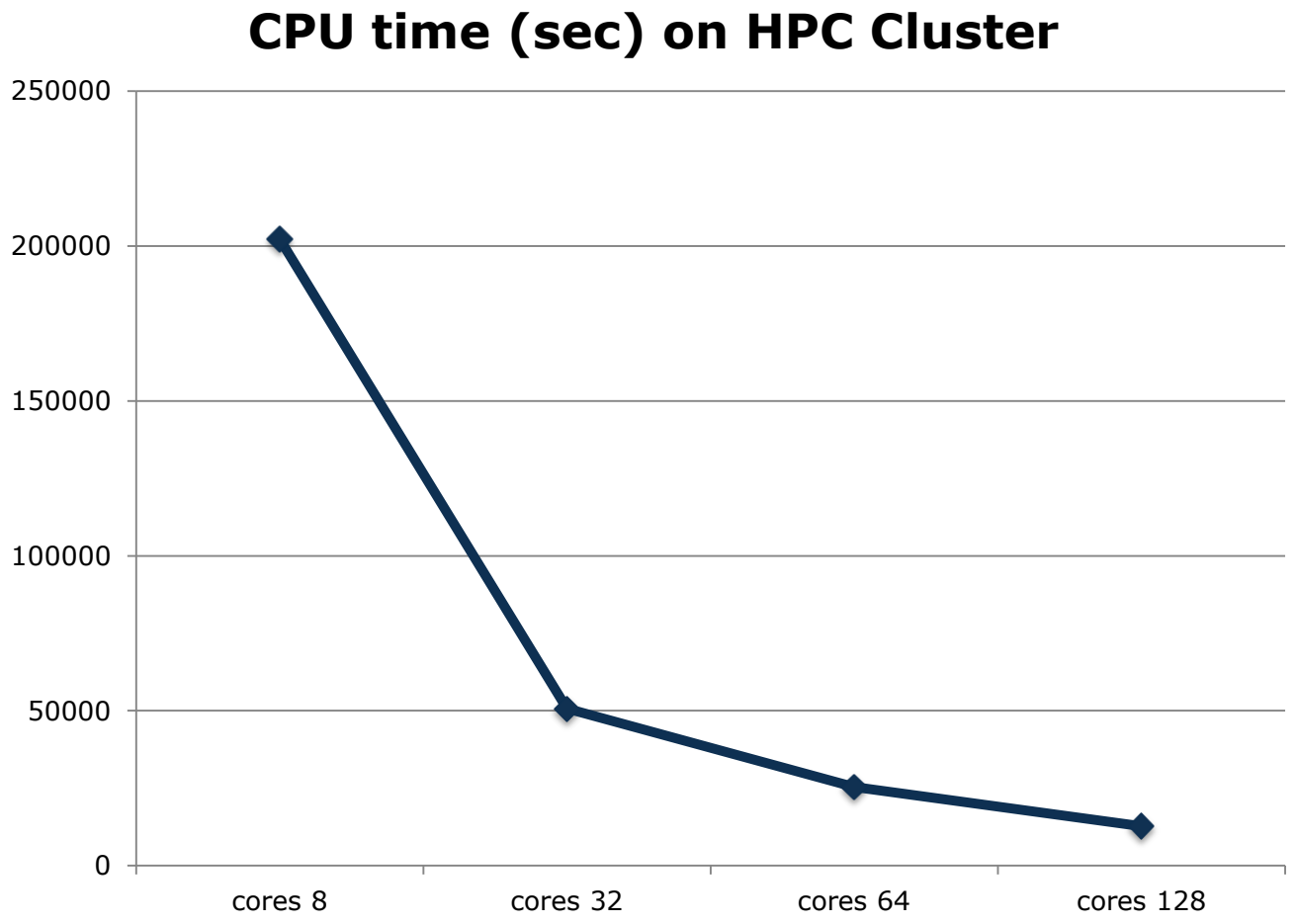


Numerical results

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

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

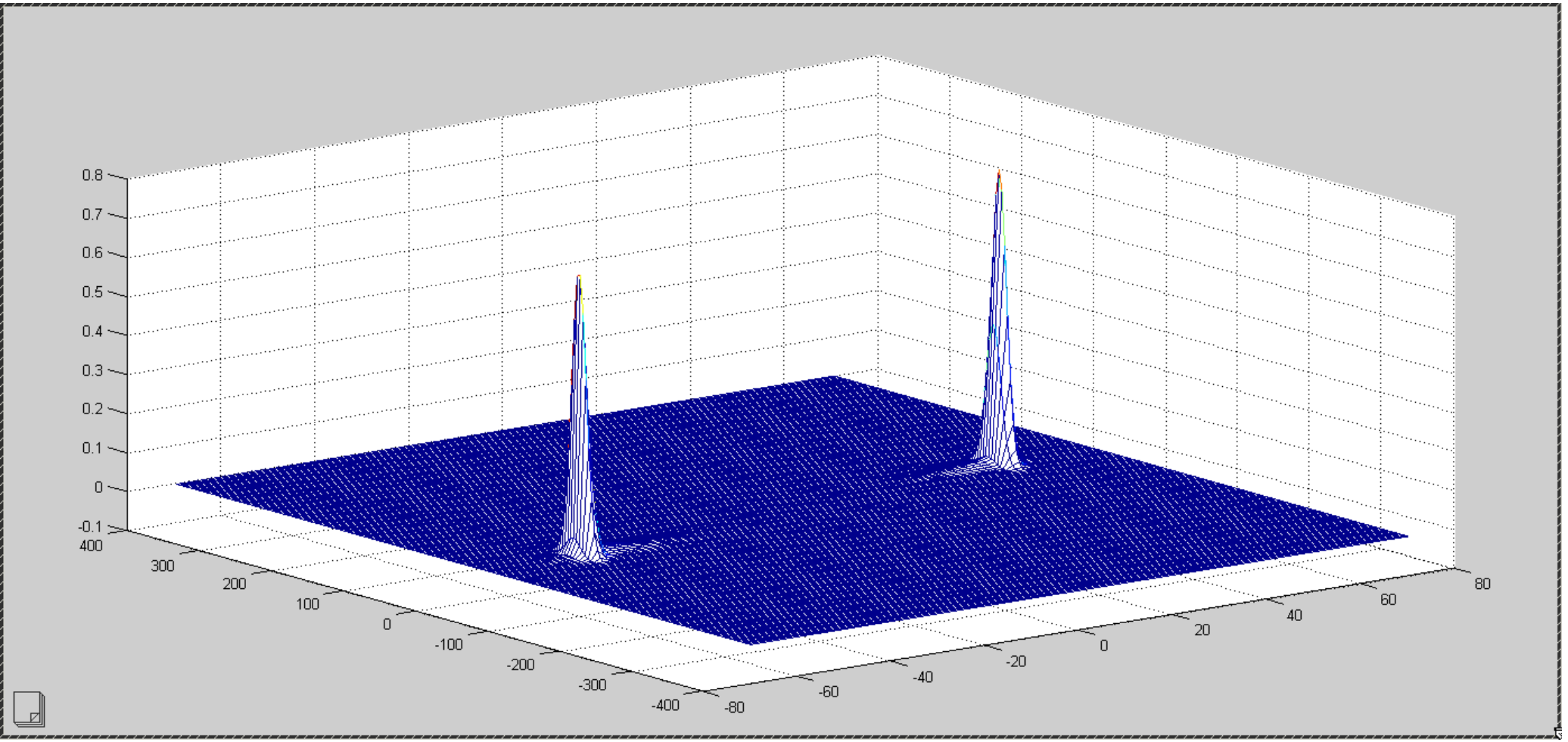
Blades/Cores	CPU Time (s)	Speed-up	Parallel Efficiency
1 x 8 = 8	202300	-	-
4 x 8 = 32	50659	3.9937	0.99834
8 x 8 = 64	25423	7.9574	0.99467
16 x 8 = 128	12735	15.8853	0.99283
Blades/Cores/ Hyperthreading	CPU Time (s)	Speed-up	Parallel Efficiency
1 x 8 x 2 = 16	148602	-	-
4 x 8 x 2 = 64	37660	3.94588	0.98647
8 x 8 x 2 =128	18957	7.83889	0.97986
16 x 8 x 2 =256	9552	15.55716	0.97232


CPU time (sec) on HPC Cluster
250000
200000
150000
100000
50000
0
cores 8
cores 32
cores 64
cores 128

Numerical results

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

Example results for the wigner function



Implementation using GPGPU

HP-SEE

High-Performance Computing Infrastructure for South East Europe's Research Communities

- The GPGPU computing uses powerful graphics cards for power and cost efficient computations.
- State-of-the-art graphics cards have large number (even thousands) 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

HP-SEE

High-Performance Computing Infrastructure for South East European Research Communities

- The Sobol sequence with Owen scrambling is generated by scrambling consequtive digits, using previous digits to generate “random trees” that serve to permute the next digits.
- For the Halton sequence we compute a list of admissible numbers for the forst 16384 primes.
- The code has been refactored to enable the main computations to be put in a GPU kernel function.
- Two kernels, one of them related to initialization of pseudo-random or quasi-random numbers is invoked once (pre-processing) and the main one is invoked repeatedly.
- One kernel invocation computes 32x256 Monte Carlo samples (trajectories).

The GPGPU-based version

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Using NVIDIA CUDA version 4.
- 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).
- Using CURAND from NVIDIA CUDA SDK for pseudorandom number generator.

The GPGPU-based version

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Observations from running the GPGPU-based version:
- Threadsize of 256 seems optimal
- significant number of divergent warps due to logical operators.
- Significant advantage for the production Tesla M2090 (Fermi) card versus previous generation essentially gaming card GTX 295.
- Around 93 % parallel efficiency achieved when 6 cards were running computations for 10^8 samples in parallel.

Numerical results of the GPGPU version

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

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

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- From our testing we concluded that hyperthreading should be used when available, production Tesla cards have much higher performance than essentially gaming cards like GTX 295, 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.