Optimization and Porting of the Path Integral Monte Carlo SPEEDUP Code to New Computing Architectures

SEE-GRID-SCI
SEE-GRID eInfrastructure for regional eScience

www.see-grid-sci.eu

SEE-GRID-SCI USER FORUM 2009 Turkey, Istanbul
09-10 December, 2009

V. Slavnić, A. Balaž, D. Stojiljković, A. Belić, A. Bogojević
Scientific Computing Laboratory
Institute of Physics Belgrade, Serbia
http://www.scl.rs/

The SEE-GRID-SCI initiative is co-funded by the European Commission under the FP7 Research Infrastructures contract no. 211338

Overview

SEE-GRID-SCI
SEE-GRID eInfrastructure for regional eScience

- Introduction
- SPEEDUP code
- Tested hardware architectures
- Results
  - Setup
  - Serial SPEEDUP code
  - MPI SPEEDUP code
  - Modified SPEEDUP code
  - Cell SPEEDUP code
- Comparison of hardware performance results
- Conclusions

Introduction

- SPEEDUP code is used for numerical studies of Quantum Mechanical systems, properties of BECs and ultra-cold atomic gases
- Porting of the code enables its use on a broader set of computing resources
- Code optimization allows us to
  - Fully utilize computing resources
  - Eliminate bottlenecks in the code
  - Use different architectures in a proper way
  - But, it must be done carefully (verification)
- Possibility of benchmarking of different hardware platforms
- Use results for planning of hardware upgrades

SPEEDUP code (1/2)

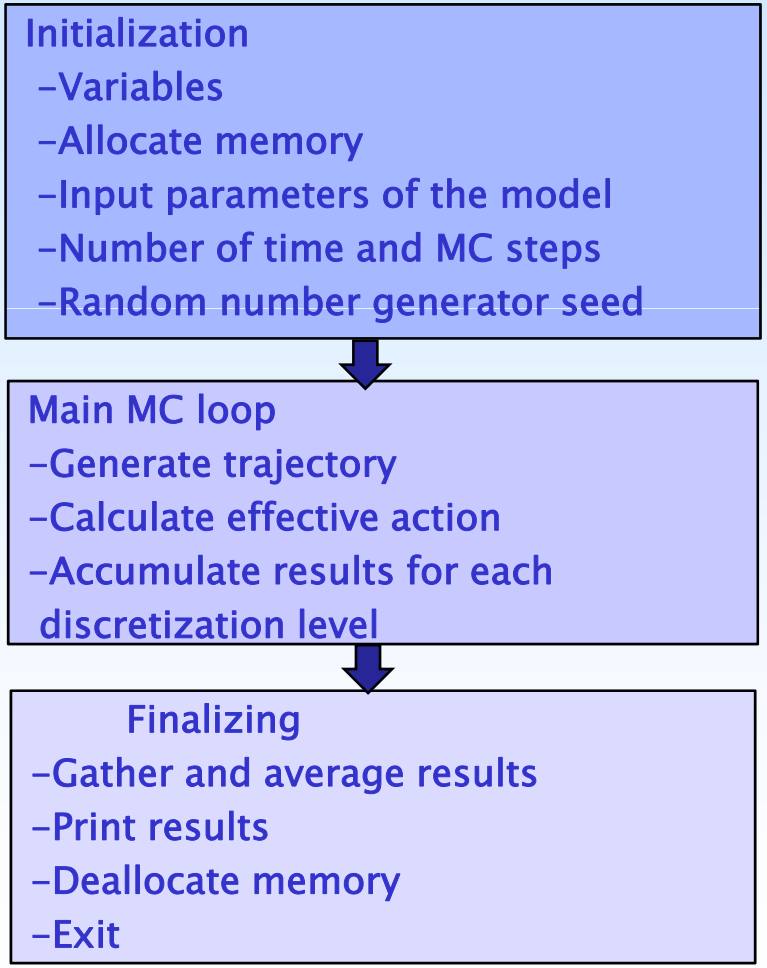
- Monte Carlo simulations are natural choice for numerical studies of relevant physical systems in the functional formalism – Path Integral Monte Carlo
- Speedup code calculates transition amplitudes using the effective action approach

$$A_N(i;f;T)=\left(\frac{1}{2\pi\varepsilon_N}\right)^{N/2}\int dq_1\ldots dq_{N-1}e^{-S_N}$$

- It is able to calculate partition functions and expectation values
- It can be also used to extract information about the low-lying energy spectra of quantum systems

SPEEDUP code (2/2)

- Algorithm:



- Good RNG is essential – we use SPRNG

Tested architectures (1/2)

- HX21XM blade Server
  - Intel Xeon based
  - 2 quadcore 5405 processors
  - ICC and GCC compilers used
- JS22 blade server
  - POWER6 based
  - 2 dualcore processors supporting multithreading and ALtiVec
  - IBM XLC/C++ and GCC compilers used

Tested architectures (2/2)

- QS22 blade server
  - Cell B/E architecture – 2 PowerXCells 8i on board
  - 1 PowerPC Processor Element (PPE)
  - 8 Synergetic Processing Elements (SPEs)
  - IBM XL C/C++ Compiler for Multicore Acceleration and GCC compilers used
- SR1625UR Intel Server System
  - Intel Xeon Nehalem based
  - 2 quadcore X5570 CPUs (Hyper-Threading)
  - ICC and GCC compilers used

Setup

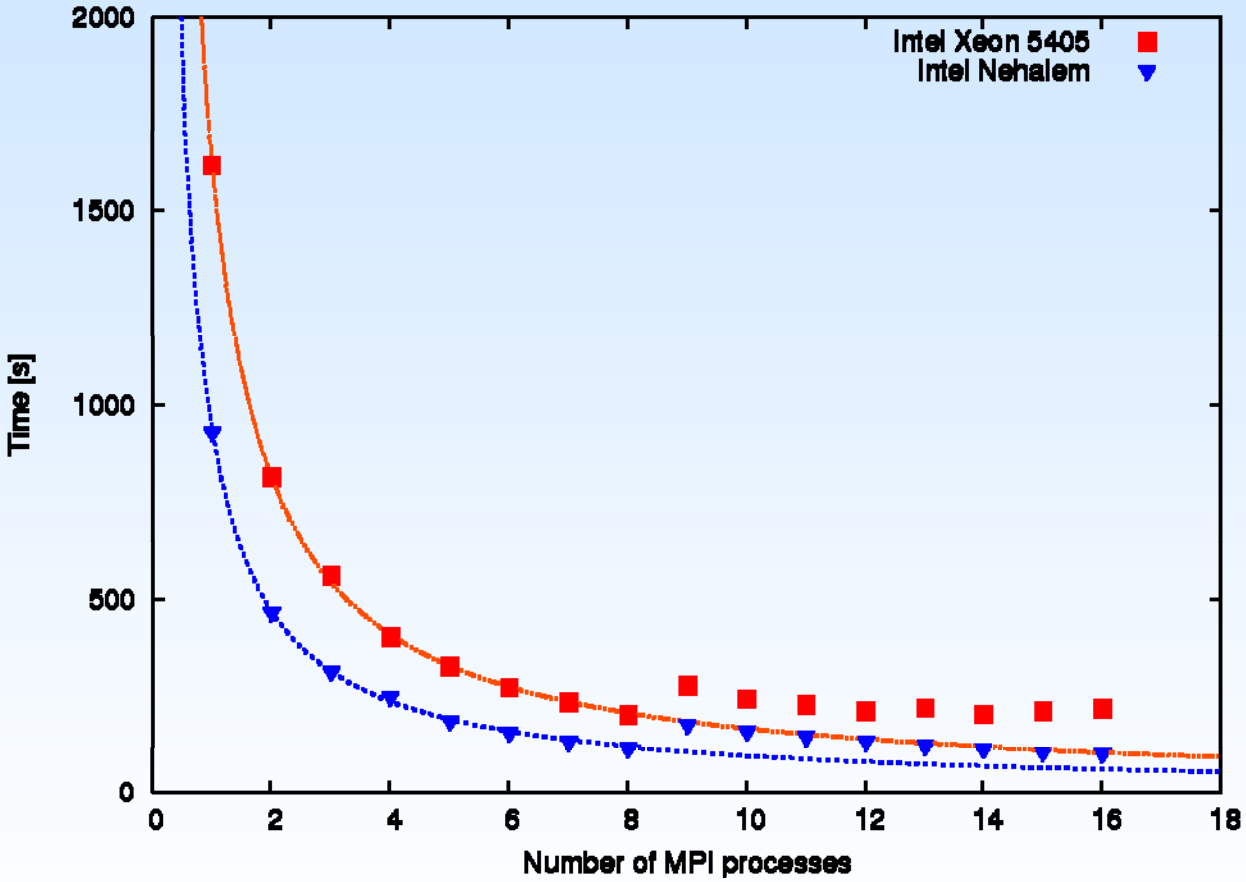
- $Nmc = 5 \times 10^6$ MC samples
- Boundary conditions for the transition amplitude
  - $q(t=0)=0$
  - $q(t=T=1)=1$
  - zero anharmonicity
  - level of effective action $p=9$ for the quartic anharmonic oscillator
- Same seed for SPRNG generator used for easy verification of the obtained results

Serial SPEEDUP results

Compiler / Platform	GCC	ICC	XLC
Intel Xeon 5405	(6280±20)*s*	(1600±20)*s*	-
Intel Nehalem	(3520±10)*s*	(920±10)*s*	-
POWER6	(8980±10)*s*	-	(1830±10)*s*
Cell	(25350±50)*s*	-	(12550±20)*s*

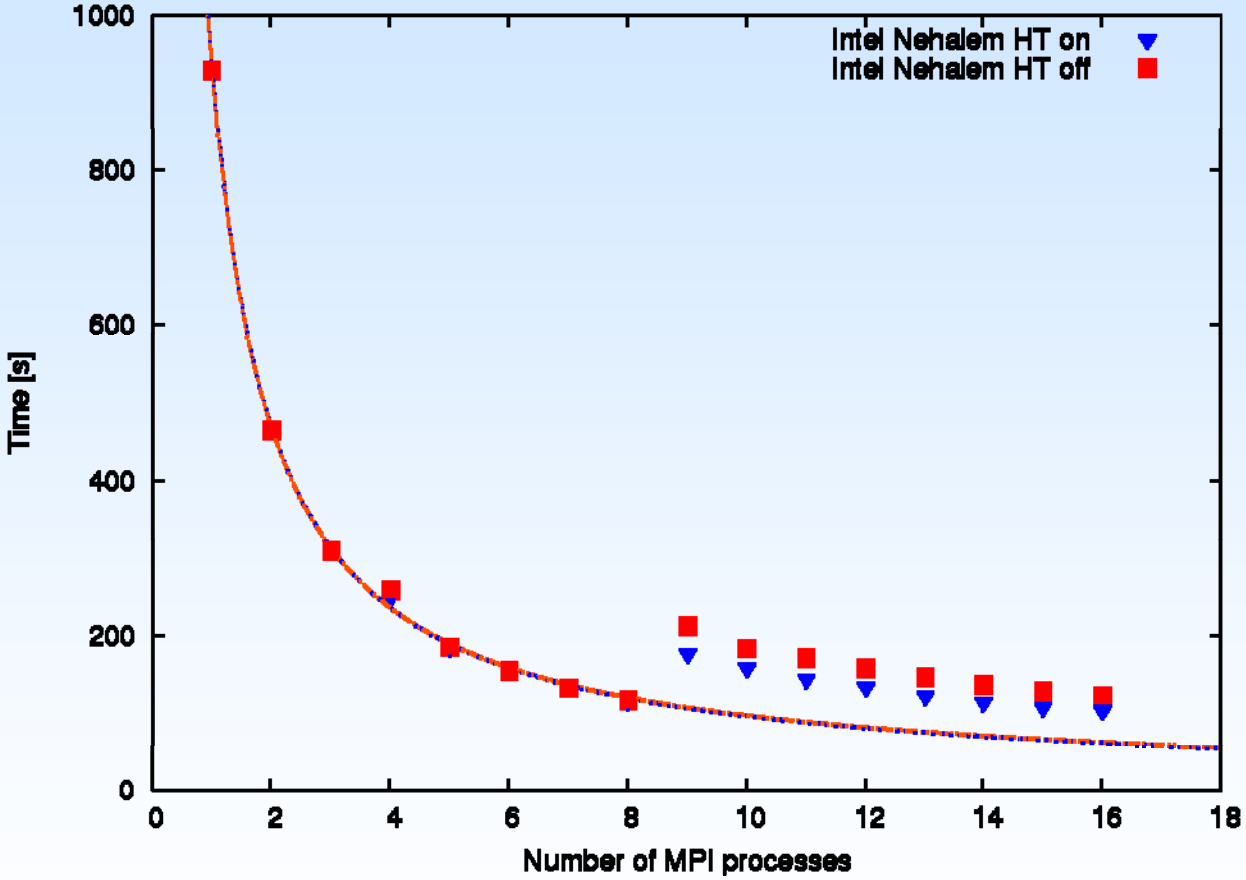
- Significant increase in the speed when platform-specific compiler is used
- Intel Nehalem performance dominates in this benchmark
- Cell is no match when only PPE is used (without the use of SPEs)

MPI SPEEDUP results(1/2)



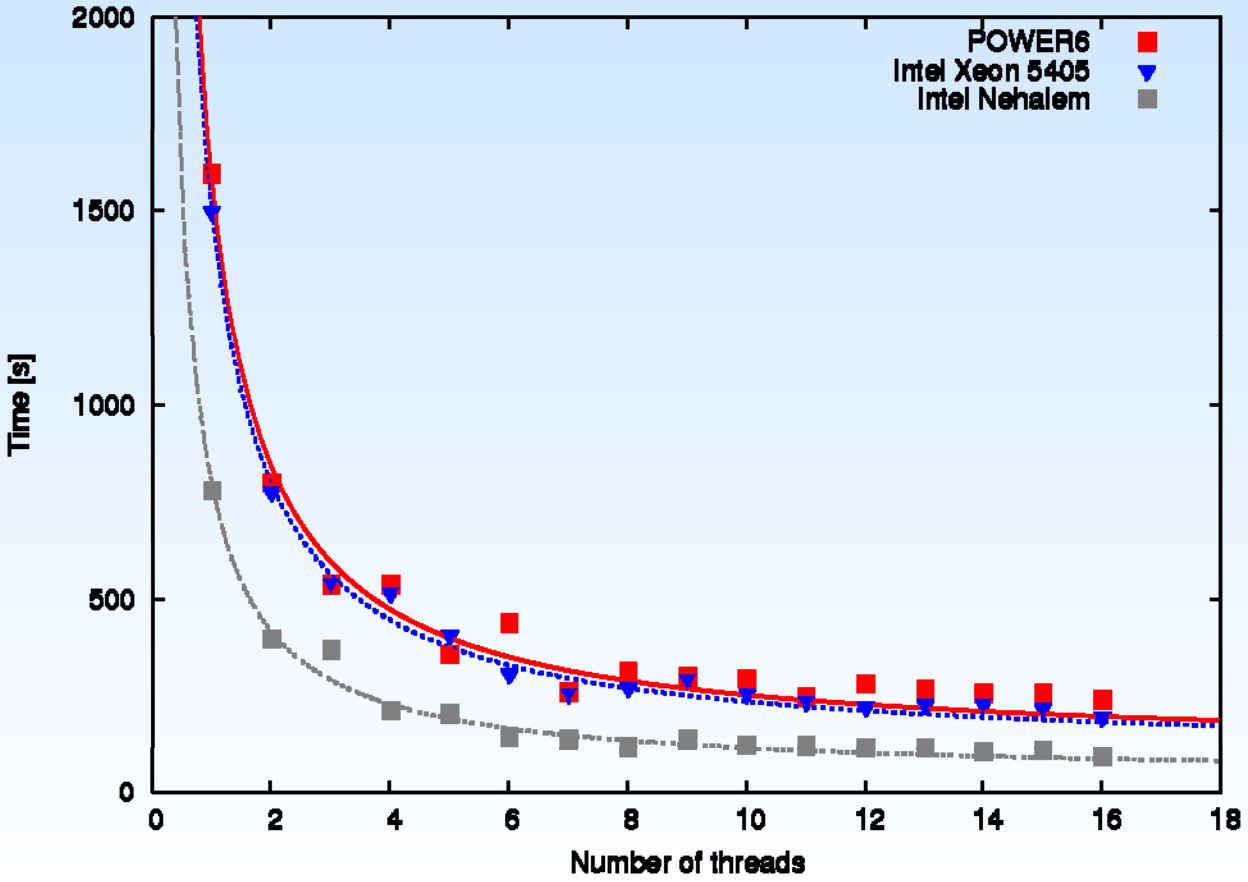
- Excellent scalability with the number of MPI processes
- MPI processes > 8 interesting behavior

MPI SPEEDUP results(2/2)



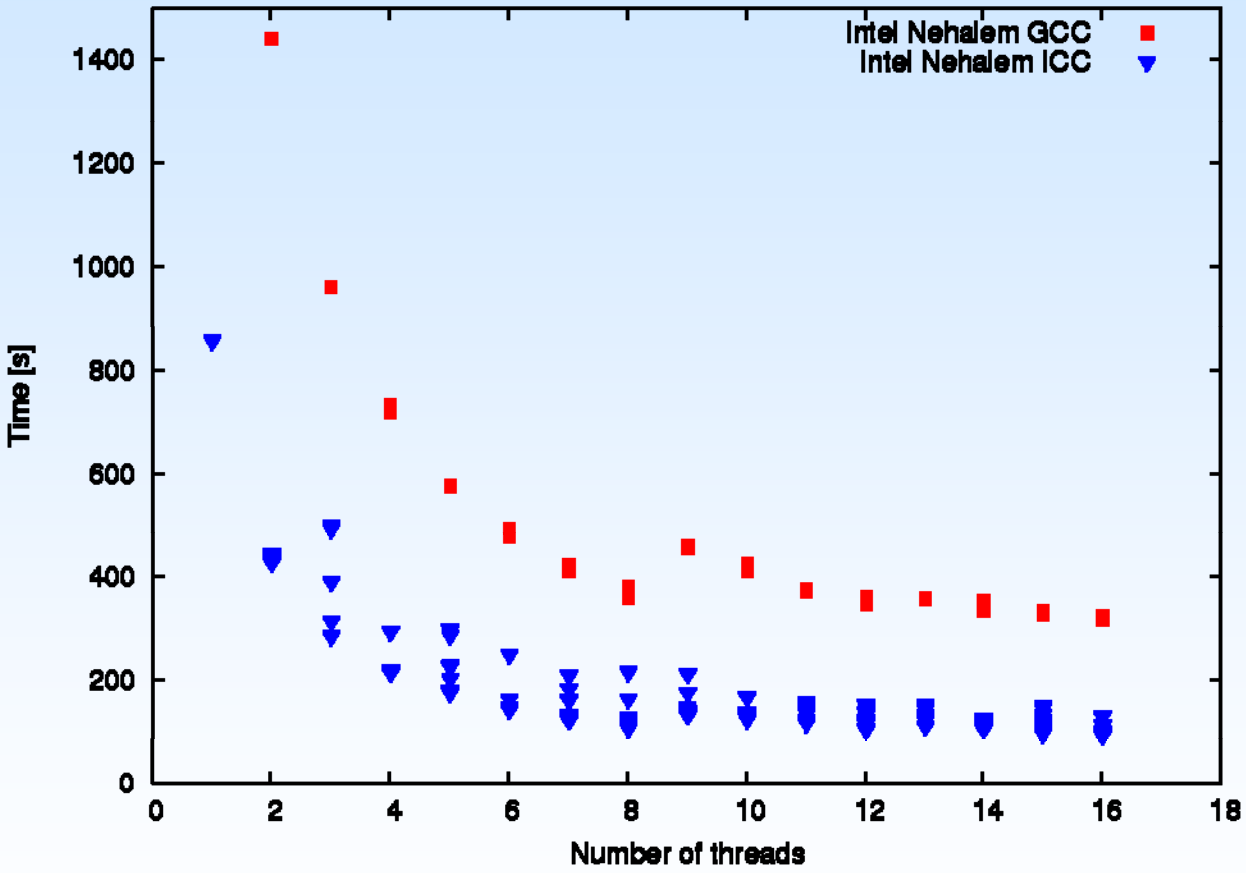
- Low Hyper-Threading performance
- Minimal execution time of 200s on Intel Xeon 5405 and 100s on Intel Hehalem

Modified SPEEDUP results(1/2)



- Implemented as a threaded version using POSIX threads
- Each thread calculates Nmc/Num_threads
- Small execution impact with Hyper-Threading

Modified SPEEDUP results(2/2)

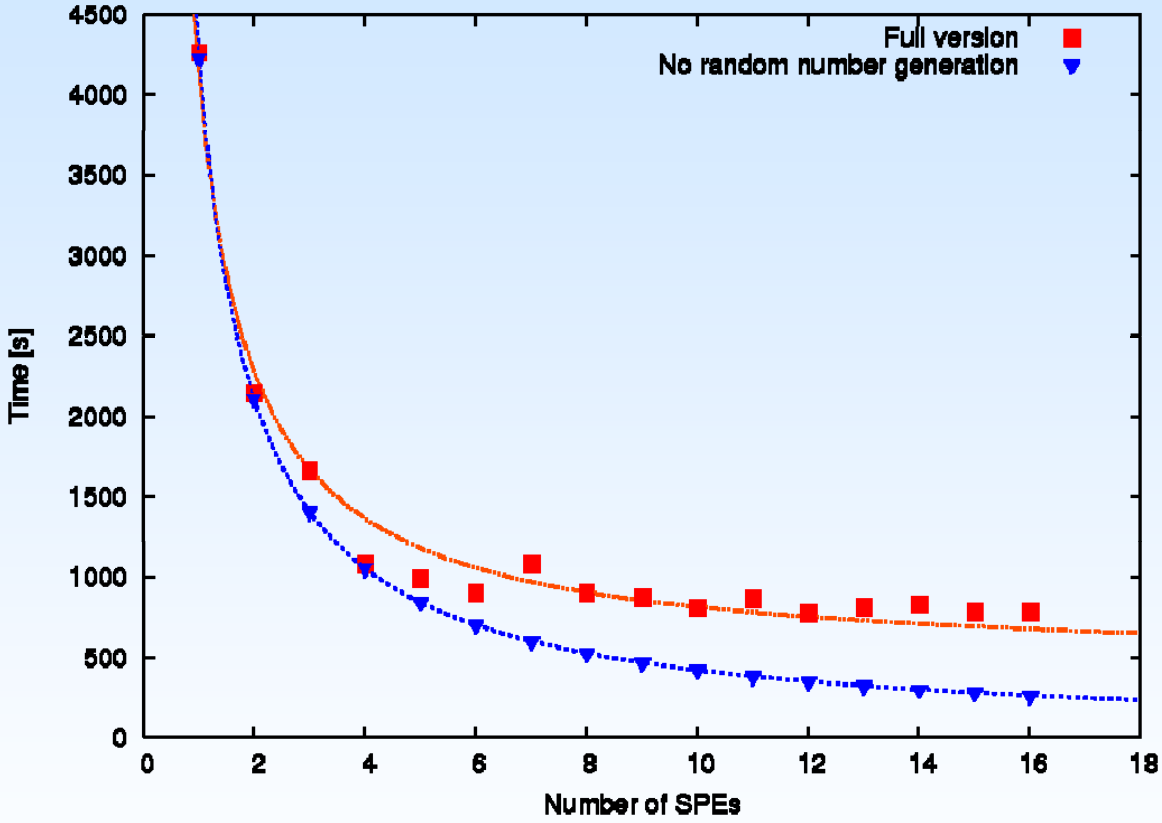


- Large scattering of times for ICC code
- Best results: Intel Xeon 5405 190s, Intel Nehalem 95s, POWER6 235s

Cell SPEEDUP results (1/3)

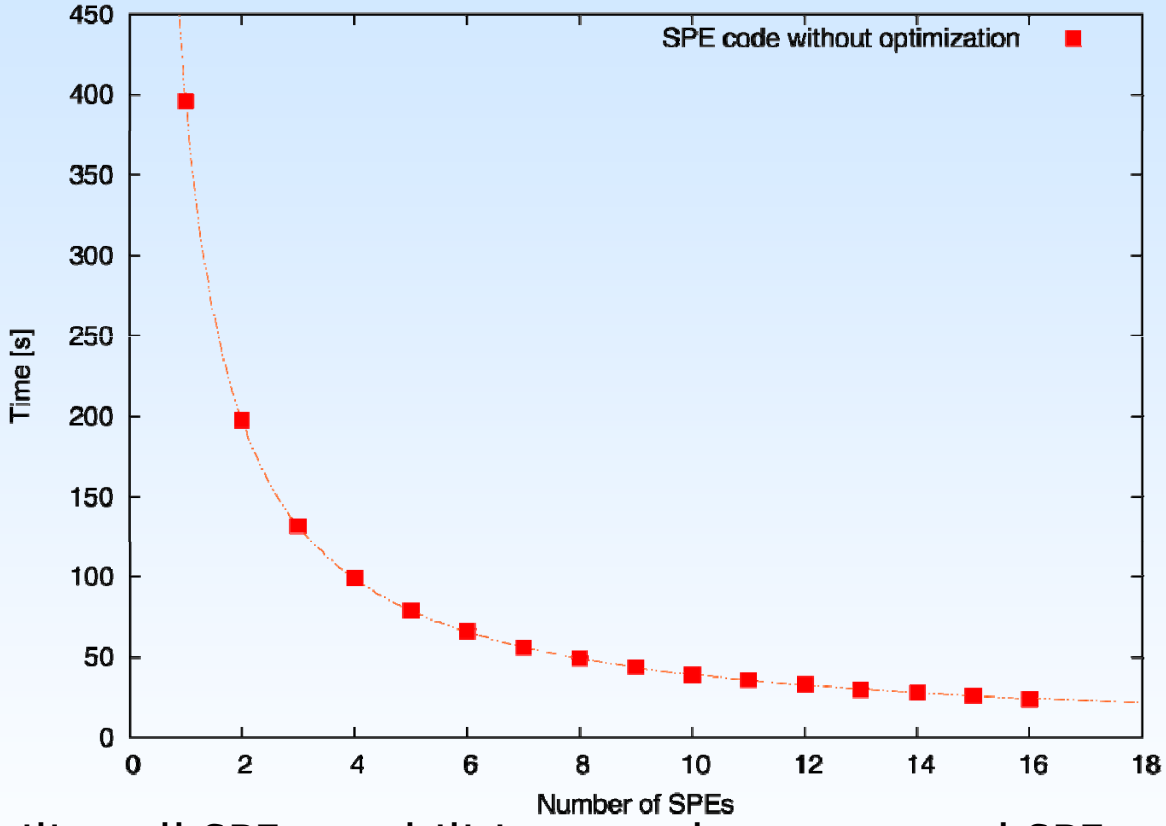
- Heterogeneity of the architecture required the slight rearrangement of the code
- Same code is executed on all SPES
- Each SPE performs Nmc/Number_of_SPEs MC steps
- No SPRNG library for SPEs
- Pthreads on PPE for control of SPEs and RNG generation
- DMA transfers of generated random trajectories from PPEs to SPEs
- Synchronization with mailbox technique

Cell SPEEDUP results (2/3)



- Saturation of the performance around 4 SPEs caused by RNG
- Communication does not have significant impact on the execution time
  - Tested with RNG only, for verification
- Test result: 770s; ideal time: 260s

Cell SPEEDUP results (3/3)



- To fully utilize all SPE capabilities, one has to extend SPE calculation time
  - Increase in the effective action level p
  - We demonstrate this by compiling the code without optimization
- Perfect scaling when PPEs have enough time for RNG

Comparison of results

Intel Xeon 5405	Intel Nehalem	POWER6	Cell	Cell ideal
190s	95s	235s	770s	260s

- Results for Intel Xeon 5405, Intel Nehalem and POWER6 are obtained using modified SPEEDUP code
- Cell ideal time corresponds to the full utilization of SPEs (estimated)
- 30% better performance of the Intel Nehalem vs Intel Xeon 5405 platform when frequencies are rescaled

Conclusions (1/2)

SEE-GRID-SCI
SEE-GRID eInfrastructure for regional eScience

- POWER6 and Intel CPUs optimization is done using threaded version of the code
- Cell platform requires more complex changes of the code
- Platform-specific compilers always give much better performance
- SPEEDUP easily optimized on the Intel platforms, with superior performance on Nehalem processors.

Conclusions (2/2)

- No significant performance improvement using Hyper-Threading technology
- Respectable level of performance with higher calculation times for Cell
- Future work: porting of SPRNG library to SPEs and implementation of platform-specific instructions (vectorization) for each tested platform