Scientific Computing with FPGAs: the JANUS machines

A case of success in Computational Physics for the study of Spin Glasses

M. Baity-Jesi, R.A. Baños, A. Cruz, L. A. Fernández, J.M. Gil-Narvion, A. Gordillo-Guerrero, D. Iñiguez, A. Maiorano, E. Marinari, V. Martin-Mayor, J. Monforte-Garcia, A. Muñoz-Sudupe, D. Navarro, S. Perez-Gaviro, J.J. Ruiz Lorenzo, B. Seoane, A. Tarancon, D. Yllanes,
Institututo de Supercomputación y Física de Sistemas Complejos (BIFI).
Zaragoza, Spain

A. Gordillo-Guerrero,
Dpto. Ingeniería Eléctrica, Electrónica y Automática
Universidad de Extremadura.
Caceres, Spain

J.J. Ruiz-Lorenzo,
Dpto. Fisica
Universidad de Extremadura.
Badajoz, Spain

M. Baity-Jesi, L. A. Fernández, V. Martin-Mayor, A. Muñoz-Sudupe,
Dpto. Física Teórica I.
Universidad Complutense
Madrid, Spain

A. Maiorano, E. Marinari, G. Parisi, B. Seoane, D. Yllanes,
Dipartimento di Fisica
Università di Roma, La Sapienza
Rome, Italy

F. Mantovani, M. Pivanti, S.F. Schifano, R. Tripiccione,
Dipartimento di Fisica
Università degli Studi di Ferrara,
Ferrara, Italy

R.A. Baños. A. Cruz, J. Monforte-Garcia, A. Tarancon,
Dpto. Física Teórica
Universidad de Zaragoza.
Zaragoza, Spain

Abstract— **Although Spin Glasses are a fundamental problem in Statistical Physics since 50 years, the gap between experiments and numerical simulations is still large. This gap has been significantly reduced by the use of programmable processors during the last years. By using them, we have developed two generations of purpose-specific machines based in FPGAs: the Janus Computers. In this paper we describe its architectures, after motivating the scientific problem. We give its performance figures and describe the current development of the next generation of the infrastructure: Janus II.**

Keywords; computer architectures, algorithms, parallel computation, FPGA

The Janus project has been partially supported by the EU (FEDER funds, No. UNZA05-33-003, MEC-DGA, Spain); by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013, ERC grant agreement no. 247328); by the MICINN (Spain) (contracts FIS2006-08533, FIS2012-35719-C02, FIS2010-16587, TEC2010-19207); by the SUMA project of INFN (Italy); by CAM (Spain); by the Junta de Extremadura (GR10158); by the Microsoft Prize 2007 and by the European Union (PIRSES-GA-2011-295302).

I. INTRODUCTION

A major challenge in condensed-matter physics is the understanding of glassy behavior, see for example [1] and [2]. Glasses are materials of the greatest industrial relevance (aviation, pharmaceuticals, automotive, etc.) that do not reach thermal equilibrium in human lifetimes. This sluggish dynamics is a major problem for the experimental and theoretical investigation of glassy behavior, placing numerical simulations at the center of the stage. Even for the most simple models, there are strong controversies regarding the low temperature state of the systems.

Spin glasses are a wide category of prototypical glassy systems, see [3]. Spin variables, taking a small set of discrete values (e.g. two states, *up* and *down*) sit at the nodes of a regular *D*-dimensional lattice, and tend to take the same value of their neighbors if the coupling defined on the edge connecting the corresponding nodes are positive (and to misalign if the coupling is negative). The coupling between neighbors is randomly chosen at the beginning of the simulation. The deceivingly simple rules governing the

evolution of a single node produce a very complex collective dynamics, especially when one deals with large lattices (order 10^6 nodes), as required to compute results that can be compared with experiments on glassy materials. In order to make this comparison with experiments, we need to follow the evolution of a large enough 3*D* lattice, say 80^3 sites, for time periods of the order of 1 second. One Monte Carlo Step (MCS) ---the update of all the 80^3 spins in the lattice--- roughly corresponds to 10^{12} seconds, so we need some 10^{12} such steps, that is ~10^{18} spin updates. Furthermore, in order to account for random couplings of the system we have to collect statistics on several (~10^2) copies of the system, adding up to ~10^{20} Monte Carlo spin updates. It is essential to realize that the correct study of glassy behavior requires that the MCS for each copy be *consecutive*. Therefore, performing this simulation program in a reasonable time frame (say, less than one year) requires a computer system able to update on average one spin in 1 picosecond or less.

At the time the project started (early 2006), available commercial CPU technology made it possible to develop simulation codes that partially exploited the available parallelism; a large optimization effort resulted in a spin-flip time of order 1 ns. A cluster of order 100 CPUs would then need around 10^9 seconds (that is, more than 30 years) to complete the simulation described above. The development of Janus, see [4], [5] and [6], marked a substantial progress: Janus deploys 256 processors built with state-of-the-art FPGAs (in 2006). The simple dynamical rules governing a single spin variable can be easily implemented as a small block of logical rules; a single FPGA hosts up to 1000 single-spin-flip engines. One Janus node has a 16 ps single spin-flip time. A simulation program as the one outlined above can be completed in just a few months, making it a viable option.

A performance increase following the Moore's law is inevitably going to finally beat Janus on spin glass applications. Nevertheless, FPGA technology improves too, and larger and faster programmable devices are continuously available on the market. These days, we are developing a new generation of FPGA-based supercomputer, named Janus II, which will be able to outperform its ancestor by at least two order of magnitude, guaranteeing for itself a long lifetime in terms of absolute performance, cost/performance and power/performance ratios.

In the rest of the paper, we briefly describe a typical spin glass model and the structure of a typical Monte Carlo simulation and discuss why its implementation is more effective on FPGAs than on conventional CPUs (and GPUs). We then describe the architecture of Janus, and give performance comparison with respect to other systems. We finally describe the new Janus II supercomputer, that is at development stage, and discuss its expected performance.

II. Simulations of Spin-Glasses

The most prototypical and simple spin-glass model is the three-dimensional Edwards-Anderson model [7] The energy of the system is defined as:

$$E = -\sum_{\langle ij \rangle} \sigma_i J_{ij} \sigma_j; \qquad (1)$$

where σ_i are the spin variables (modeling magnetic moments at atomic lattice sites) taking values +1 (spin up) and -1 (spin down), sitting at the nodes of a three-dimensional cubic lattice of linear size *L*. J_{ij} are the strengths of the interaction (couplings) along the edges connecting nearest-neighbor nodes. A positive J_{ij} favors alignment of the spins of the two neighboring nodes while a negative value favors misalignment. The sum in (1) spans all pairs of nearest neighbors. For the values in the set of couplings J_{ij}, we consider the case in which the couplings are +1 or -1 with probability 0.5 (binary model). A set of $3L^3$ J_{ij} couplings is a *sample* of the system; the couplings in a sample remain fixed in time.

The *local energy* of a single spin, say σ_k at site *k*, is determined by the interaction with its six neighbors only:

$$\epsilon(\sigma_k) = -\sigma_k \phi_k; \qquad (2)$$

$$\phi_k = \sum_{j=k\pm x,k\pm y,k\pm z} J_{kj}\sigma_j\,. \qquad (3)$$

The *local field* Φ_k is determined only by the nearest neighbors of σ_k.

The probabilities of the spin to be *up* or *down* can be computed from the Boltzmann-Gibbs distribution:

$$P(\sigma_k = \pm 1) = \frac{\exp\left[\pm\beta\phi_k\right]}{\exp\left[\beta\phi_k\right] + \exp\left[-\beta\phi_k\right]}\,, \qquad (4)$$

where $\beta=1/T$ is the *inverse temperature*. This defines the *Heat-Bath* algorithm: at any given time, we may decide if the spin σ_k is up or down by comparing the probability to be up with a (pseudo-) random number ρ extracted uniformly in the interval *[0,1)*.

The recipe for the simulation of the dynamics of a single sample of the model (1) is then as follows:

1 - Extract the complete J_{ij} configuration.

2 - Extract the initial $\{\sigma_i\}$ configuration (each spin can be up or down with equal probability).

3 - Begin a trial spin-flip: pick a site *k* at random, each site with equal probability.

4 - Compute the local field $\varepsilon(\sigma_k)$ from (3) and the spin up probability $P(+1)$ from (4).

5 - Pick a uniformly distributed pseudo-random number $0 \le \rho < 1$ from your favorite generator.

6 - If $\rho < P(+1)$, then put $\sigma_k = +1$, otherwise put $\sigma_k = -1$; end of the trial spin-flip.

7 - Repeat from step 3 above as many times as needed.

A MCS in this scheme is a number of trial spin-flip equal to the number of sites in the lattice.

When one performs a huge number of MCSs, the statistical properties of the observables that one measures (averages over all sites and averages over MCSs) do not depend anymore on

the particular order in which the algorithm visits each lattice sites. It turns out that this property is independent of any particular lexicographic order we could impose on the site-visiting scheme. This brings substantial simplification and makes room for a very efficient exploitation of the internal parallelism.

In addition, another degree of replication is needed because some interesting properties of the system comes out when comparing two independent simulations of the same sample, starting from independent initial spin configurations.

Usually each sample must be simulated twice at least; the two identical copies with independent histories are called replicas.

Traditional architectures, at the time the project started, were poor in exploiting the internal parallelism. The generation of several high-quality random numbers (one per spin) was the principal bottle-neck in the computation. Traditional CPU are in fact tailored to manage complex basic computations (integer and floating-point arithmetic) on (relatively) large data sets (32- and 64-bit words). The ideal spin glass machine would be instead more similar to an *application specific* GPU, with data paths tailored to perform the specific sequence of logical operations, a control structure shared by a number of cores larger than in state-of-the-art GPUs, data storage on on-chip memory only and a memory controller optimized for typical access patterns as required by the described algorithm.

Janus is essentially an array of hundreds of thousand of properly tailored small computational cores performing simulations of systems of interacting discrete variables on regular structures. Nevertheless, Janus is not ``only`` an application specific facility, performing well on a wide range of applications but with outstanding performances on spin glasses.

III. The Janus Architecture

A Janus board (see Fig.1) contains 16+1 FPGA-based components: 16 SPs (*Simulating Processors*) and an IOP (the *Input-Output Processor*). A board needs an host PC to be operated. The SP is the unit in which we exploit internal parallelism, efficiently implementing the algorithm described before. External parallelism is usually obtained by farming SPs (16 per board) and by driving more boards. In its full configuration, Janus is a stack of 16 boards, totaling 256 SPs. All the devices inside the Janus Board can be managed through the IOP, which is connected to the host PCs via a gigabit Ethernet interface. The host PC runs a standard Linux operating system, and a set of specific C libraries allow a user to get through the link to and from the IOP.

We choose the Xilinx Virtex-4 LX200 FPGA as the reconfigurable device for IOPs and SPs. The main clock in the board is 62.5 MHz and it is distributed to all IOP and SP devices. Such a conservative choice has been useful to guarantee mapping of our application to very dense firmware codes (we reached near 95\% resource occupation for our heaviest applications).

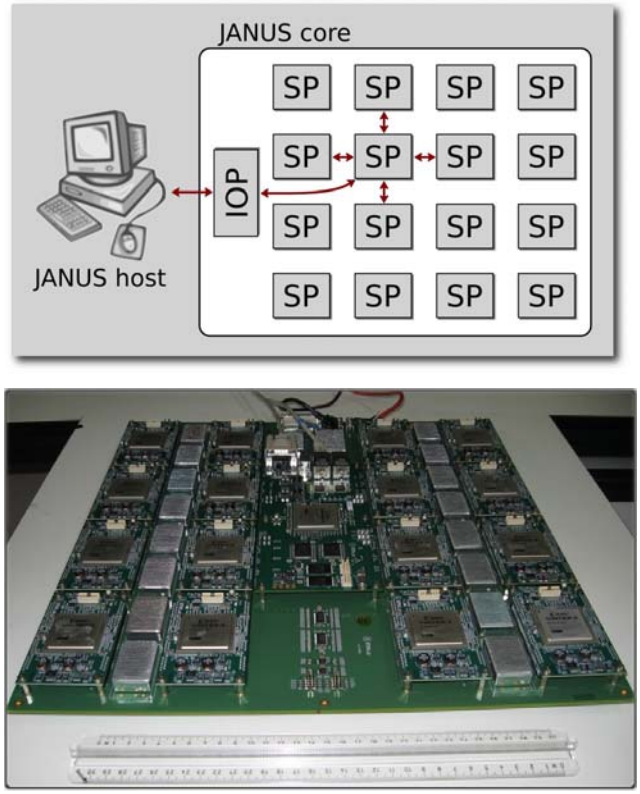

Figure 1. Top: a schematic view of the Janus board internal and external connections. Bottom: a real photo of a working Janus board.

The IOP card also include 8MB of static memory, a PROM programming device (to load the IOP's FPGA firmware on power-up) and some I/O interfaces: two Gigabit Ethernet and a serial link (useful for debugging purposes).

The SPs are the computational nodes of Janus. Each SP may operate independently or together with other SPs, depending on the firmware loaded by the user program running in the host PCs. The user application consists basically of two layers: a firmware layer, loaded on the SPs, implements the many cores performing the computational tasks as described in Section II. A software layer, the *Janus Operating System* (JOS), comprises a program running as a background process in the host PC (*josd*); it performs the abstraction and acts as a job queue manager.

Development of a Janus application consists in fact of two tasks: programming the firmware for the SPs (in a hardware description language, VHDL is our choice), complying with the I/O requirements in the IOP, and programming the mid-level library functions that, based on the {\tt josd} interface to user application, implement the chosen protocol for communication with the SPs.

A. A Sample Spin Glass Application Implementation

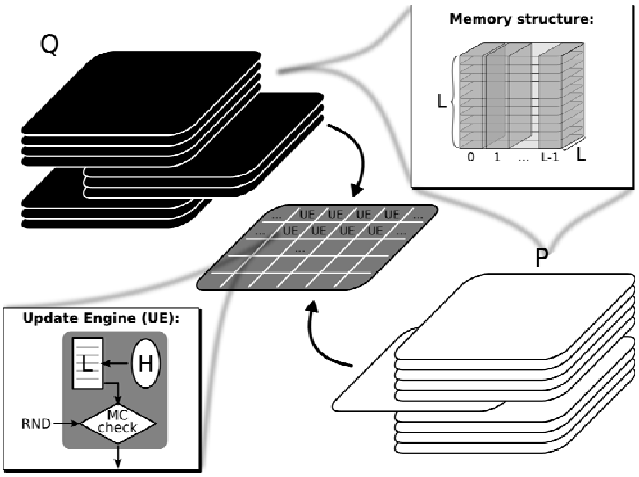

Figure 2. Diagram of the SP firmware operation. In the UE, the logic block *H* computes the local field, and use it as an address to the probability Look-Up Table *L*; the *MC check* block compares probability value with an incoming pseudo-random number and returns the new spin value.

We try to exploit all FPGA resources to achieve best performance. The Virtex-4 LX200 FPGA by Xilinx comes with many small RAM blocks that we can logically combine and stack to reproduce the 3*D* array of spins on a lattice of size *L* (*L* 2*D* memories with width *L* and depth *L*). In addition, we *have* to simulate two replicas at least as mentioned in Section II. Then, we consider two replicas of a single disorder sample and divide them in black and white sites in a checkerboard scheme; then, we arrange all black spins of one replica together with all white spins of the other replica in the same 3*D* memory structure. We end up with two mixed 3*D* memory structures (we call them structure *P* and structure *Q*); each spin in one structure has neighbors only in the other structure: no spins in the same structure are neighbors of each other in the physical lattices. Now, we can read or write an *L*-wide word from each memory of the 3*D* structure *P* per clock cycle; it turns out that having *L* memories, we can read an entire plane of the 3*D* memory *P*, update it and write back to memory at the next clock cycle; we only need three planes of *neighbors* from the 3*D* memory structure *Q*; if we update planes from, say, bottom to top; of the three planes of neighbors in *Q* needed to update a plane in *P*, two of them will be neighbors also for the subsequent plane of the structure *P*. At regime, spanning the whole 3*D* memory structure, only one read and at most one write is needed from each RAM block in order to update an entire plane of the *P* structure. We can then update an entire plane of L^2 spins per clock cycle. Besides, we instantiate identical 3*D* memory structures to store all the necessary coupling constants, to be fetched with the same rate as the spins in the *P* structure. We then arrange a set of L^2 identical *update engines* (UE); each UE receives in input the six neighbors of a single spin and combinatorially computes the local field; the pre-computed and constant-in-time spin-up probabilities, normalized to 32-bit unsigned integers, are stored in a Look-Up Table; each UE contains its own table and accesses it independently; the UE also receives a 32-bit word of random bits; a comparison between the addressed 32-bit probability and the 32-bit random number is performed and the new value of the updated spin is returned.

Once the whole *P* structure is updated, the controlling state machine interchanges the roles of *P* and *Q*, and the update of the *Q* structure starts with the same procedure described above. When the *Q* structure is updated, a Monte Carlo step is complete. The whole machinery is depicted in Fig.2

A key advantage of the FPGA implementation is that we can generate pseudo-random numbers concurrently with the rest of the computation and feed them to UEs at a proper rate.

For more details on the efficient implementation of the previous algorithm we refer the reader to (4).

It is usually possible to place up to 1024 UEs in a single FPGA, thus sustaining an update time of 16 ps per spin at 62.5 MHz clock speed. We have described the implementation on a quite simple (but still very interesting) spin-glass model, we successfully programmed and used the Janus computer to simulate various glassy systems. The interested reader can find in the works (8), (9), (10) and (11), an exhaustive survey.

IV. JANUS PERFORMANCES

In the case of a CPU implementation we have two approaches: the *Asynchronous Multi-Spin Coding* (AMSC) approach, consist on filling bit positions in a long word with spins (or couplings) belonging to the same site of independent samples. On the other hand, in the *Synchronous Multi-Spin Coding* (SMSC) approach, one fills the bits with spin variables (and couplings) taken from a single sample, choosing of course a set of spins that can be updated simultaneously.

In what follows we consider the single spin-update-time (SUT) as the unit of performance. On an SP running 1024 updates per clock cycle, at 62.5 MHz clock cycle, the SUT is 16 ps.

We performed several performance measures on a wide range of many-core systems (Cell Broadband Engine, 4-core Nehalem Intel CPU Xeon 5560, Tesla C1060 GP-GPU), and some partial tests on a 8-core Intel Sandy Bridge processor (Xeon E5-2680), see~\cite{Gui:10}.We compare Janus performances to mixed AMSC-SMSC implementations, reporting on two kind of SUT measures: the time needed for completing the simulation divided by the total amount of trial spin-flips needed to update a single sample (single-system SUT): this has to be compared with the full SMSC implementation in a single Janus SP; the time needed for completing the simulation divided by the total amount of trial spin-flips needed to update all the simulated samples (global SUT): we compare this to the performance of a Janus board, in which one may consider both AMSC and SMSC levels of parallelism. Results are summarized in Table 1.

The figures we show for the GPU processor deserve a further comment. Floating-point application usually get large advantages by the use of GPUs, but this is not the case for our sample application; in facts, the Tesla C1060 does not really

TABLE I. Comparison of Janus performance to some commercial processor: Intel Nehalem (dual socket board with two 4-core Xeon 5560), Tesla C1060 and Intel Sandy-Bridge (dual socket board with two 8-core Xeon E5-2680). In the upper part, the single-system spin update time, to be compared with the SMSC implementation of a single SP of Janus. In the lower part, the global spin-update time, to be compared to the global spin-update time of a whole Janus board. The number of independent systems simulated in parallel in the AMSC scheme is shown in parentheses

single-system SUT (ns/spin)						
L	Janus SP	I-NH (8 Cores)	CBE (8-SPE)	CBE (16-SPE)	Tesla C1060	I-SB (16 cores)
16	0.063	0.98	0.83	1.17	–	–
32	0.016	0.26	0.40	0.26	1.24	0.37
48	0.021	0.34	0.48	0.25	1.10	0.23
64	0.016	0.20	0.29	0.15	0.72	0.12
80	0.020	0.34	0.82	1.03	0.88	0.17
96	0.027	0.20	0.42	0.41	0.86	0.09
128	–	0.20	0.24	0.12	0.64	0.09
global SUT (ns/spin)						
L	Janus	I-NH (8 Cores)	CBE (8-SPE)	CBE (16-SPE)	Tesla C1060	I-SB (16 cores)
16	0.004 (16)	0.031 (32)	0.052 (16)	0.073 (16)	–	–
32	0.001 (16)	0.032 (8)	0.050 (8)	0.032 (8)	0.31 (4)	0.048 (8)
48	0.0013 (16)	0.021 (16)	0.030 (8)	0.016 (16)	0.27 (4)	0.015(16)
64	0.001 (16)	0.025 (8)	0.072 (4)	0.037 (4)	0.18 (4)	0.015 (8)
80	0.0013 (16)	0.021 (16)	0.051 (16)	0.064 (16)	0.22 (4)	0.011 (16)
96	0.0017 (16)	0.025 (8)	0.052 (8)	0.051 (8)	0.21 (4)	0.012 (8)
128	–	0.025 (8)	0.120 (2)	0.060 (2)	0.16 (4)	0.011 (8)

outperform other standard architectures, and keeps one order of magnitude less performing than Janus hardware. GPU are more effective on applications with reduced memory accesses; the computational kernel of our application requires few operations: the local field of a single spin, for instance, can be computed with 6 XORs and 5 sums on a set of 12 variables (6 neighbor spins and 6 couplings). The GPU performances are then limited by memory-bandwidth despite peak performances are one order of magnitude larger than competing multi-core architectures.

We see then that recent standard architectures will indeed fill the gap with Janus. Still, the Janus computer, based on six-year-old technology, remains architecture of choice for spin glass simulations. The figures we presented show that Janus is capable of performing a given simulation in 10 times shorter wall-clock-time. Still, FPGA technology has followed its course and we expect that today state-of-the-art FPGA based processors would outperform the Janus SPs.

V. JANUS II

The next generation of the Janus supercomputer will respect the general architecture of its predecessor, with many improvements made possible by technology advances in the last years.

We plan the following main improvements:

- Latest-generation FPGA devices.

- A tighter coupling between the IOP and the host PC.

- A faster and more flexible communication between IOP and SPs inside one board and across boards.

Janus II is a again cluster of processing boards. Each board is a set of 16 SPs and an IOP as in Janus, built on a processing board (PB). The PB provides all electrical links and connectors to arrange the SP grid into a 4x4x1 3*D* toroidal network: the added third dimension with respect to the Janus board allow for communication between SPs on adjacent boards. The PB also provides the 125 MHz master clock to all devices.

The IOP features the largest increase in complexity with respect to other Janus components. It will integrate all of the previous IOP functionality and will integrate the host PC. A Computer-On-Module (COM) express board plugs onto the IOP piggy-back module. The COM express system will have Sandy-Bridge class CPUand at least 4 GB DDR3 memory. A Solid State Disk will be connected to the COM express via SATA links. The core of the IOP will of course be an FPGA performing all data-routing and control. We select the Xilinx Virtex-7 XC7VX485T. The FPGA processor will export a 8x PCI Express Gen 2 link to the COM module and provide enough high-speed serial links (GTX) to all the SPs on the board and to IOPs in other boards. The IOP interfaces to the world via an Infiniband adapter; there will be service connections also: two Gigabit Ethernet channels and two serial interfaces. The IOP of course provides also the programming interface for on-the-fly configuration of the SPs' FPGAs.

The SP also features one Xilinx Virtex-7 XC7VX485T. At variance with the previous Janus SP, it will also host some DDR3 DRAM. It will have direct connections to neighboring SPs in the 3*D* logical toroidal network, physically implemented with high-speed serial transceivers in the FPGA (5 GB/s transfer rate minimum).

The fast interconnection network and the new FPGA features provide large margins for performance improvements. Janus II features FPGAs with more than twice the logic, 6 times more distributed memory and 6 times more block RAM (see Table 2) than the largest Virtex-4 devices.

TABLE II: Resource comparison between FPGAs in Janus (Xilinx XC4VLX200) and Janus II (Xilink XC7VX485T).

device	logic cells	distributed RAM	block RAM
XC4VLX200	200448	1392 Kb	6048 Kb
XC7VX485T	485760	8175 Kb	37080 Kb

The available logic permits a factor 2 increase in terms of number of updates per clock-cycle. Another factor 2 comes from the faster clock of the system (in Janus it was 62.5 MHz). Please note that we consider running even at faster speed inside the SPs, and that we are conservatively supposing that our application will run at the rate given by the master clock. The main difference with Janus will be the high-speed direct connections between the SPs in the 3D mesh: each link is about five times faster than each link in the 2D SP mesh of a Janus board.

Taking in consideration the sample application described in Section II, available memory and connection speed allow for simulating two replicas of a single sample of a 3D spin glass on a lattice of linear size L=1200 dividing it into sublattices of 300x300x150 among the 128 SPs of 8 adjacent boards. With a sustained update rate of 2000 spins per clock cycles on 300x300 planes inside each SPs, the most demanding bandwidth would be the border spins transfer through the links connecting SPs in different boards, requiring 2000 bits transfered in 150 cycles (before they become necessary to start the next update sequence in the neighbor SP) corresponding to 3.3 Gb/s.

Such implementation would then be capable of an update rate of 256000 spins per clock-cycle, corresponding to a single-system SUT of 32×10^{-15} (and the global SUT is just half that amount in a complete configuration with 16 Janus II boards).A simulation program as the one described at the end of section 4 (two replicas, 10^{11} Monte Carlo steps) with only 16 samples, would obtain relevant results in about one year.

In a single SP the single-system SUT would be around 4 times better than in Janus. With only two SPs, we could allocate two replicas of a L=150 system and simulate them with a 4500 updates per spin rate and complete the task in a month. On a farm of 16 8-cores Sandy-Bridge processors it would take almost two years (assuming the measured performance for a L=128 system from Table 1 applies).

The Janus II computer then establish again a significant gap with commercial processors.

VI. Conclusions

We have described two Janus generations of supercomputer for Spin Glass application. In the second generation we expect a significant improvement with respect to the previous one, due to progresses in FPGA technology and some major architectural design modifications.

We expect Janus II to be a durable facility; in Fig. 3 we compare best performance values taken from Table 1. If the trend in Intel performance growth should continue, and if the estimate we provide for the Janus II computer holds, the latter has an advantage of almost four years, which is enough to ensure a significant output of scientific production. If the trend continues by the usual Moore's law starting form the Sandy-Bridge performances, Janus II will again remain unrivaled for some years.

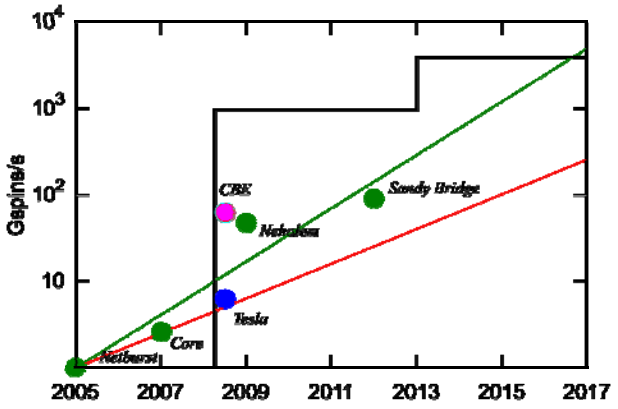

Figure 3. Performance growth in time; each point correspond to the time we actually get a code running on a specific architecture. Green points are the Intel series (one CPU); the red point corresponds to the CBE and the blue point is the Tesla GPU. The black line are performances of one board of the Janus machines (Janus II is expected starting full scientific production by the end of 2013). The red line represents the performance growth predicted by a Moore's Law with doubling performance each 18 months. The green line is the performance growth trend extracted by considering only our data points for the Intel series,

References

[1] C.Angell, "Formation of glasses from liquids and biopolymers". Science, 267, pp.1924-1935 (1995),.

[2] P. Debenedetti, "Metastable liquids". Princeton University Press, Princeton (1997).

[3] M. Mezard, G. Parisi, and M. Virasoro, "Spin Glass Theory and Beyond". World Scientific, Singapore (1987).

[4] F. Belletti, et al., "Simulating spin systems on IANUS, an FPGA-based computer". Computer Physics Communications, 178, 208 (2008).

[5] F. Belletti, et al., "JANUS: an FPGA-based Sys-tem for high performance scientic computing". Computing in Science & Engineering, 11, pp. 48-58 (2009).

[6] M. Baity-Jesi, et al., "Reconfigurable computing for Monte Carlo simulations: Results and prospects of the Janus project". The European Physical Journal Special Topics, 210, pp.33-51 (2012).

[7] S. Edwards, and P. Anderson, "Theory of spin glasses". J. Phys. F: Metal Phys., 5, pp.965-974 (1975).

[8] R.A. Baños, et al., "Critical behavior of three-dimensional disordered potts models with many states". J. Stat. Mech., 2010(5), P05002 (2010).

[9] R.A. Baños, et al., "Static versus dynamic hetero-geneities in the d = 3 Edwards-Anderson-ising spin glass". Phys. Rev. Lett., 105, 177202 (2010).

[10] R.A. Baños, et al., "Sample-to-sample fuctuations of the overlap distributions in the three-dimensional Edwards-Anderson spin glass" Phys. Rev. B, 84, 174209 (2011).

[11] A. Cruz, et al., "Spin glass phase in the four-states three-dimensional potts model". Phys. Rev. B, 79, 184408 (2009).