Janus, an FPGA-Based System for High-Performance Scientific Computing

David Yllanes for the Janus Collaboration[1]

Dep. Física Teórica I, Universidad Complutense de Madrid

Computer Simulations on GPU
Mainz, June 1, 2011

[1] R. A. Banos, A. Cruz, L.A. Fernandez, J. M. Gil-Narvion, A. Gordillo-Guerrero, M. Guidetti, A. Maiorano, F. Mantovani, E. Marinari, V. Martin-Mayor, J. Monforte-Garcia, A. Muñoz Sudupe, D. Navarro, G. Parisi, S. Perez-Gaviro, J.J. Ruiz-Lorenzo, S.F. Schifano, B. Seoane, A. Tarancon, R. Tripiccione and D. Yllanes

Esquema

Spin glasses

- Spin glasses: random, mixed-interacting, magnetic system. Random, yet cooperative, freezing of spins at a temperature T_c.
- Disorder + Frustration $\Longrightarrow$ Complex behaviour.

Spin glasses

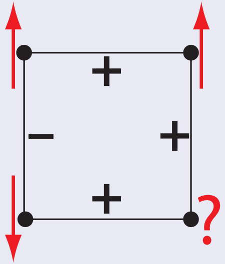
- Spin glasses: random, mixed-interacting, magnetic system. Random, yet cooperative, freezing of spins at a temperature T_c.
- Disorder + Frustration $\Longrightarrow$ Complex behaviour.

Edwards-Anderson model

- Generally used:

$$\mathcal{H} = -\sum_{\langle i,j \rangle} J_{ij} s_i s_j, \quad s_i = \pm 1$$

- Quenched and random interactions: $J_{ij} = \pm 1$ with 50% probability.



Spin glasses

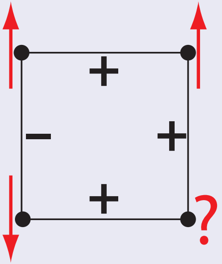
- Spin glasses: random, mixed-interacting, magnetic system. Random, yet cooperative, freezing of spins at a temperature T_c.
- Disorder + Frustration $\Longrightarrow$ Complex behaviour.

Edwards-Anderson model

- Generally used:

$$\mathcal{H} = -\sum_{\langle i,j \rangle} J_{ij} s_i s_j, \quad s_i = \pm 1$$

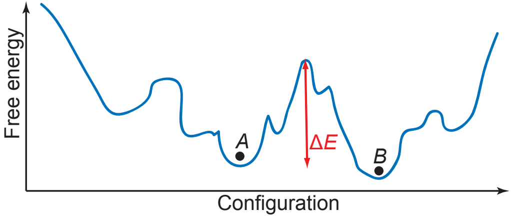
- Quenched and random interactions: $J_{ij} = \pm 1$ with 50% probability.



- Experimental system:
  - Dilute magnetic atoms in non-magnetic material (e.g. Mn in Cu).
  - RKKY interaction: sign oscillates with distance $\Rightarrow$ frustration.

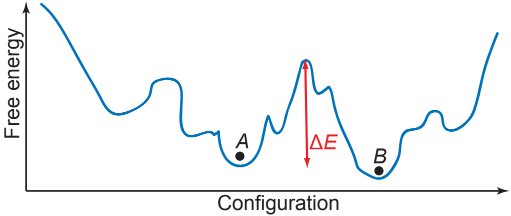
Free-energy landscapes and slow dynamics

- The dynamics is very slow below the critical temperature.
- The system is trapped for long times in free-energy valleys.



Free-energy landscapes and slow dynamics

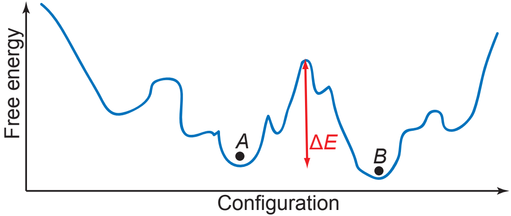
- The dynamics is very slow below the critical temperature.
- The system is trapped for long times in free-energy valleys.



- The experimental system never actually reaches equilibrium.

Free-energy landscapes and slow dynamics

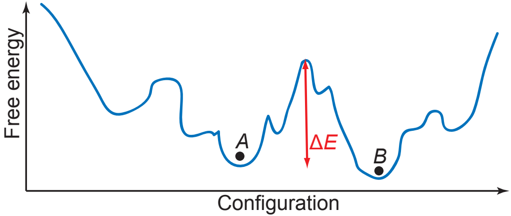
- The dynamics is very slow below the critical temperature.
- The system is trapped for long times in free-energy valleys.



- The experimental system never actually reaches equilibrium.
- Interesting experiments on non-equilibrium effects (aging).

Free-energy landscapes and slow dynamics

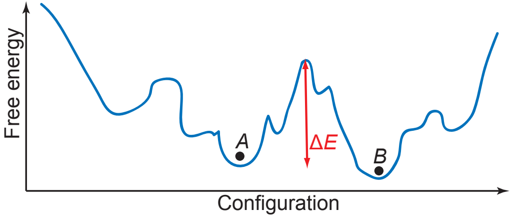
- The dynamics is very slow below the critical temperature.
- The system is trapped for long times in free-energy valleys.



- The experimental system never actually reaches equilibrium.
- Interesting experiments on non-equilibrium effects (aging).
- Relevant to other systems with complex free-energy landscapes
  - Protein folding
  - Error correcting codes
  - Vortex glasses in high-T_c superconductors

Free-energy landscapes and slow dynamics

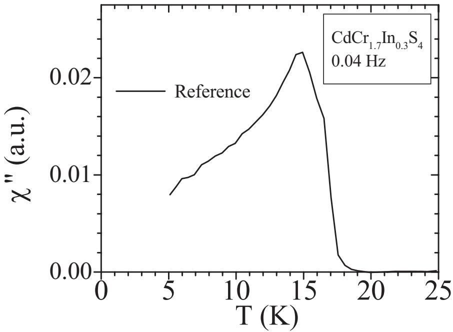
- The dynamics is very slow below the critical temperature.
- The system is trapped for long times in free-energy valleys.



- The experimental system never actually reaches equilibrium.
- Interesting experiments on non-equilibrium effects (aging).
- Relevant to other systems with complex free-energy landscapes
  - Protein folding
  - Error correcting codes
  - Vortex glasses in high-T_c superconductors
- Spin glasses are paradigmatic problems:
  - Amenable to precise experimental investigation
  - Simple theoretical models are faithful to the physics

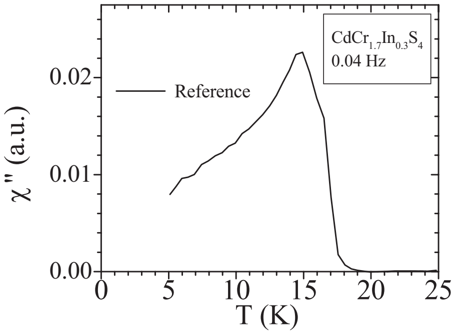
Complex dynamical behaviour

- Experiments on spin glasses have observed very complex behaviour.
- Example: memory effects [Jonason et al., PRL **81**, 3243 (1998)]



Complex dynamical behaviour

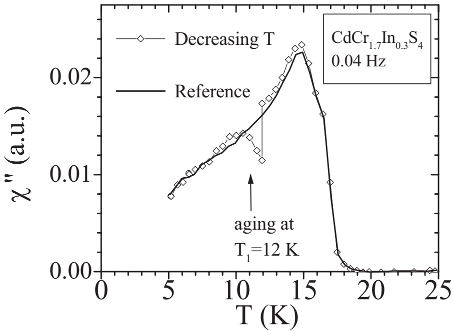
- Experiments on spin glasses have observed very complex behaviour.
- Example: memory effects [Jonason et al., PRL **81**, 3243 (1998)]



1. Cool (and reheat) the system at a slow rate

Complex dynamical behaviour

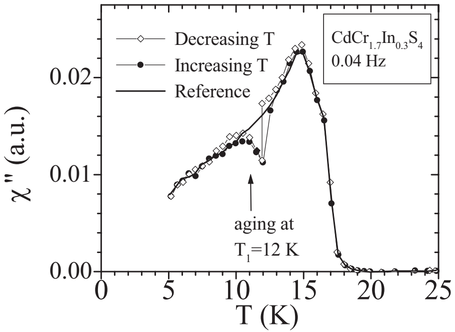
- Experiments on spin glasses have observed very complex behaviour.
- Example: memory effects [Jonason et al., PRL **81**, 3243 (1998)]



1. Cool (and reheat) the system at a slow rate
2. Cool again, but this time stop (age) at T_1.
3. Resume the cooling, the system 'rejuvenates'

Complex dynamical behaviour

- Experiments on spin glasses have observed very complex behaviour.
- Example: memory effects [Jonason et al., PRL **81**, 3243 (1998)]



1. Cool (and reheat) the system at a slow rate
2. Cool again, but this time stop (age) at T_1.
3. Resume the cooling, the system 'rejuvenates'
4. Reheat without stopping. The system has memory of the aging.

The direct quench protocol

- We want to understand spin-glass dynamics thoroughly.
- The first step is understanding isothermal aging

The direct quench protocol

- We want to understand spin-glass dynamics thoroughly.
- The first step is understanding isothermal aging
- Simple experimental protocol: direct quench
  1. Start at time $t = 0$ at $T \to \infty$ (random configuration).

The direct quench protocol

- We want to understand spin-glass dynamics thoroughly.
- The first step is understanding isothermal aging
- Simple experimental protocol: direct quench
  1. Start at time $t = 0$ at $T \to \infty$ (random configuration).
  2. Cool the system instantaneously to $T < T_c$

The direct quench protocol

- We want to understand spin-glass dynamics thoroughly.
- The first step is understanding isothermal aging
- Simple experimental protocol: direct quench
  1. Start at time $t = 0$ at $T \to \infty$ (random configuration).
  2. Cool the system instantaneously to $T < T_c$
  3. Let the system age for a waiting time t_w.

The direct quench protocol

- We want to understand spin-glass dynamics thoroughly.
- The first step is understanding isothermal aging
- Simple experimental protocol: direct quench
  1. Start at time $t = 0$ at $T \to \infty$ (random configuration).
  2. Cool the system instantaneously to $T < T_c$
  3. Let the system age for a waiting time t_w.
  4. Measure the system properties at time $t + t_w$.

The direct quench protocol

- We want to understand spin-glass dynamics thoroughly.
- The first step is understanding isothermal aging
- Simple experimental protocol: direct quench
  1. Start at time $t = 0$ at $T \to \infty$ (random configuration).
  2. Cool the system instantaneously to $T < T_c$
  3. Let the system age for a waiting time t_w.
  4. Measure the system properties at time $t + t_w$.
- Straightforward to simulate (heat-bath dynamics, e.g., reproduces the physical evolution).

The direct quench protocol

- We want to understand spin-glass dynamics thoroughly.
- The first step is understanding isothermal aging
- Simple experimental protocol: direct quench
  1. Start at time $t = 0$ at $T \to \infty$ (random configuration).
  2. Cool the system instantaneously to $T < T_c$
  3. Let the system age for a waiting time t_w.
  4. Measure the system properties at time $t + t_w$.
- Straightforward to simulate (heat-bath dynamics, e.g., reproduces the physical evolution).
- Problem 1:
  - Experimentally relevant times: a few seconds to a few hours.
  - 1 Monte Carlo Sweep (MCS) $\longleftrightarrow 10^{-12}$ s

The direct quench protocol

- We want to understand spin-glass dynamics thoroughly.
- The first step is understanding isothermal aging
- Simple experimental protocol: direct quench
  1. Start at time $t = 0$ at $T \to \infty$ (random configuration).
  2. Cool the system instantaneously to $T < T_c$
  3. Let the system age for a waiting time t_w.
  4. Measure the system properties at time $t + t_w$.
- Straightforward to simulate (heat-bath dynamics, e.g., reproduces the physical evolution).
- Problem 1:
  - Experimentally relevant times: a few seconds to a few hours.
  - 1 Monte Carlo Sweep (MCS) $\longleftrightarrow 10^{-12}$ s
- Problem 2:
  - We need the system to remain off-equilibrium for very long times $\Rightarrow$ simulate very large lattices.

Optimisation possibilities (I)

- Since we want to emulate the physical evolution, we cannot use optimised dynamics.
- We can, however, optimise our heat-bath implementation through parallelisation.

Asynchronous multi-spin coding (ASMSC)

- We need to consider a disorder average, i.e., simulate several samples (choices of $\{J_{ij}\}$).

Optimisation possibilities (I)

- Since we want to emulate the physical evolution, we cannot use optimised dynamics.
- We can, however, optimise our heat-bath implementation through parallelisation.

Asynchronous multi-spin coding (ASMSC)

- We need to consider a disorder average, i.e., simulate several samples (choices of $\{J_{ij}\}$).
- Spins are $\pm 1 \implies$ we only need one bit for each.

Optimisation possibilities (I)

- Since we want to emulate the physical evolution, we cannot use optimised dynamics.
- We can, however, optimise our heat-bath implementation through parallelisation.

Asynchronous multi-spin coding (ASMSC)

- We need to consider a disorder average, i.e., simulate several samples (choices of $\{J_{ij}\}$).
- Spins are $\pm 1 \implies$ we only need one bit for each.
- For each site in the lattice, we can fit the corresponding spins from 64–128 samples in a single integer.

Optimisation possibilities (I)

- Since we want to emulate the physical evolution, we cannot use optimised dynamics.
- We can, however, optimise our heat-bath implementation through parallelisation.

Asynchronous multi-spin coding (ASMSC)

- We need to consider a disorder average, i.e., simulate several samples (choices of $\{J_{ij}\}$).
- Spins are $\pm 1 \implies$ we only need one bit for each.
- For each site in the lattice, we can fit the corresponding spins from 64–128 samples in a single integer.
- Update them all at the same time, using logical operations.

Optimisation possibilities (I)

- Since we want to emulate the physical evolution, we cannot use optimised dynamics.
- We can, however, optimise our heat-bath implementation through parallelisation.

Asynchronous multi-spin coding (ASMSC)

- We need to consider a disorder average, i.e., simulate several samples (choices of $\{J_{ij}\}$).
- Spins are $\pm 1 \implies$ we only need one bit for each.
- For each site in the lattice, we can fit the corresponding spins from 64–128 samples in a single integer.
- Update them all at the same time, using logical operations.
- Only one random number per site, shared for all samples.

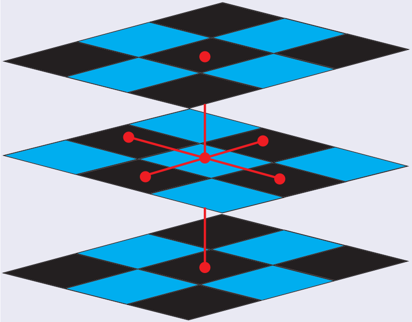
Optimisation possibilities (II)

- ASMSC
  - Good for equilibrium, where one needs many samples.
  - However, does not reduce wall-clock time.
  - Out of equilibrium we need fewer samples (self-averaging), but much longer times.

Optimisation possibilities (II)

- ASMSC
  - Good for equilibrium, where one needs many samples.
  - However, does not reduce wall-clock time.
  - Out of equilibrium we need fewer samples (self-averaging), but much longer times.

Synchronous multi-spin coding (SMSC)

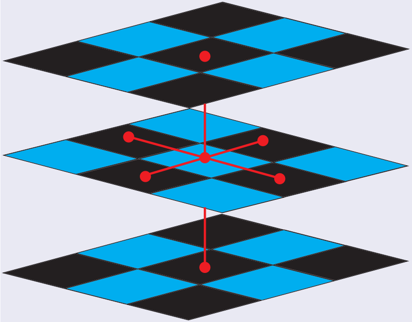


- Parallelise within each sample

Optimisation possibilities (II)

- ASMSC
  - Good for equilibrium, where one needs many samples.
  - However, does not reduce wall-clock time.
  - Out of equilibrium we need fewer samples (self-averaging), but much longer times.

Synchronous multi-spin coding (SMSC)

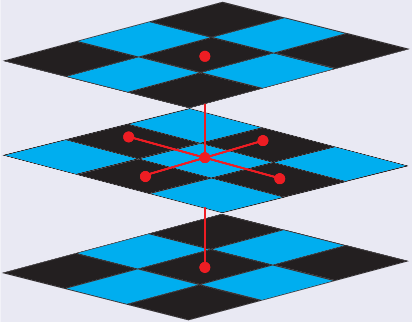


- Parallelise within each sample
- This is allowed by the local nature of the interactions

Optimisation possibilities (II)

- ASMSC
  - Good for equilibrium, where one needs many samples.
  - However, does not reduce wall-clock time.
  - Out of equilibrium we need fewer samples (self-averaging), but much longer times.

Synchronous multi-spin coding (SMSC)

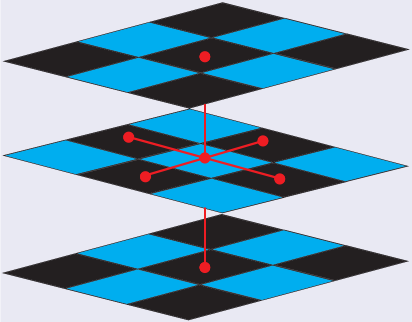


- Parallelise within each sample
- This is allowed by the local nature of the interactions
- We divide the lattice in a checkerboard scheme, all sites of the same colour can be updated simultaneously

Optimisation possibilities (II)

- ASMSC
  - Good for equilibrium, where one needs many samples.
  - However, does not reduce wall-clock time.
  - Out of equilibrium we need fewer samples (self-averaging), but much longer times.

Synchronous multi-spin coding (SMSC)



- Parallelise within each sample
- This is allowed by the local nature of the interactions
- We divide the lattice in a checkerboard scheme, all sites of the same colour can be updated simultaneously
- Now we need one random number per spin

Optimisation limits (in a PC)

- SMSC has the potential to accelerate each sample and reduce the wall-clock.

Optimisation limits (in a PC)

- SMSC has the potential to accelerate each sample and reduce the wall-clock.
- However, PCs are not equipped to handle it:
  - CPUs optimised for long data words, but we need
    - Operations on single bits (the spins)
    - Variables that only appear in a small number of states (local field, needed to compute the update probability)

Optimisation limits (in a PC)

- SMSC has the potential to accelerate each sample and reduce the wall-clock.
- However, PCs are not equipped to handle it:
  - CPUs optimised for long data words, but we need
    - Operations on single bits (the spins)
    - Variables that only appear in a small number of states (local field, needed to compute the update probability)
  - Memory architecture: the processor cannot gather all the necessary information quickly enough.

Optimisation limits (in a PC)

- SMSC has the potential to accelerate each sample and reduce the wall-clock.
- However, PCs are not equipped to handle it:
  - CPUs optimised for long data words, but we need
    - Operations on single bits (the spins)
    - Variables that only appear in a small number of states (local field, needed to compute the update probability)
  - Memory architecture: the processor cannot gather all the necessary information quickly enough.
- Sharing the simulation across several CPUs does not work, due to communication limitations (small chunks of data, but extremely often)

Optimisation limits (in a PC)

- SMSC has the potential to accelerate each sample and reduce the wall-clock.
- However, PCs are not equipped to handle it:
  - CPUs optimised for long data words, but we need
    - Operations on single bits (the spins)
    - Variables that only appear in a small number of states (local field, needed to compute the update probability)
  - Memory architecture: the processor cannot gather all the necessary information quickly enough.
- Sharing the simulation across several CPUs does not work, due to communication limitations (small chunks of data, but extremely often)
- We need a different kind of architecture

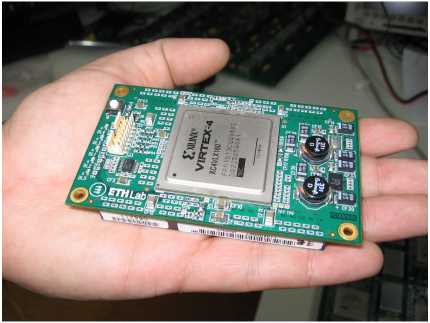
Esquema

Our solution: an FPGA-based architecture

- The Janus solution: replace CPUs by FPGAs.

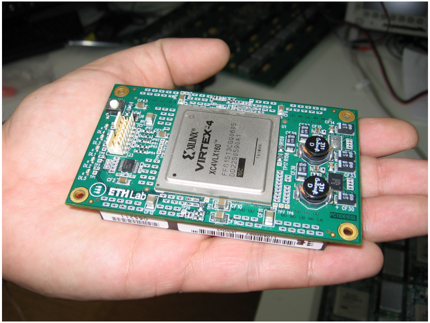
Our solution: an FPGA-based architecture

- The Janus solution: replace CPUs by FPGAs.
- FPGAs
  - Offer a large number of logic resources



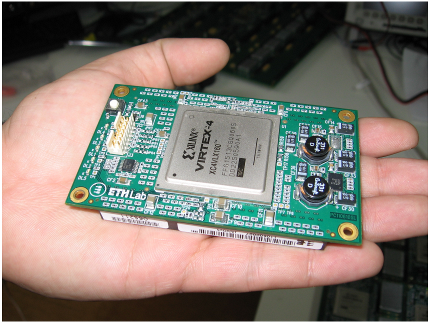
Our solution: an FPGA-based architecture

- The Janus solution: replace CPUs by FPGAs.
- FPGAs
  - Offer a large number of logic resources
  - Can be divided into many Update Engines, each accessing with no latency the necessary information to update one spin.



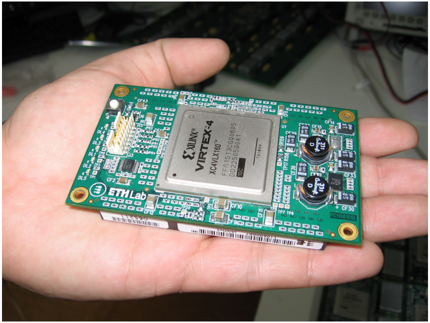
Our solution: an FPGA-based architecture

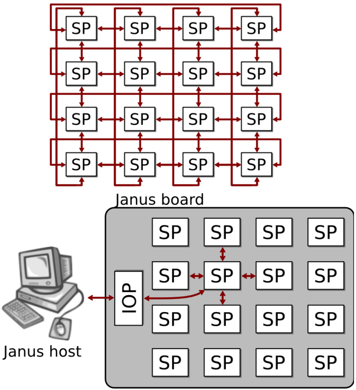
- The Janus solution: replace CPUs by FPGAs.
- FPGAs
  - Offer a large number of logic resources
  - Can be divided into many Update Engines, each accessing with no latency the necessary information to update one spin.
  - Are reconfigurable.



Our solution: an FPGA-based architecture

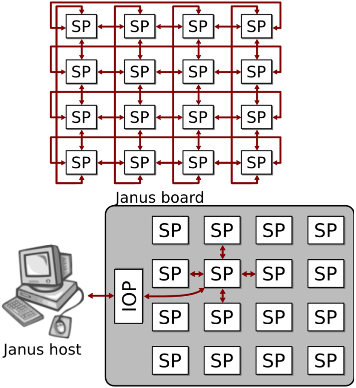
- The Janus solution: replace CPUs by FPGAs.
- FPGAs
  - Offer a large number of logic resources
  - Can be divided into many Update Engines, each accessing with no latency the necessary information to update one spin.
  - Are reconfigurable.
  - Permit a modular approach





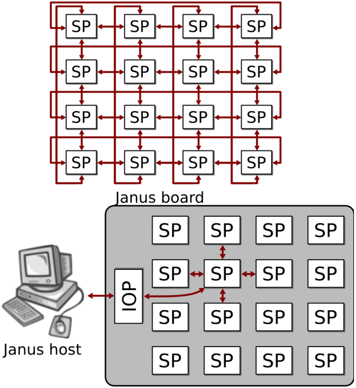
- We group 4×4 FPGA-based processors (SPs) in a $2D$ grid (a board).
- Data links among nearest neighbours

The Janus machine



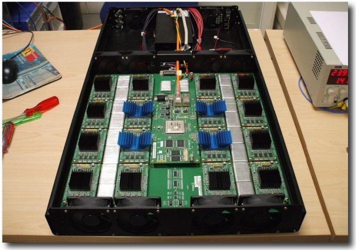
- We group 4×4 FPGA-based processors (SPs) in a $2D$ grid (a board).
- Data links among nearest neighbours
- One I/O processor per board (IOP)

The Janus machine



- We group 4×4 FPGA-based processors (SPs) in a $2D$ grid (a board).
- Data links among nearest neighbours
- One I/O processor per board (IOP)
- A standard PC (Janus host) for each 2 boards

The Janus machine



- We group 4×4 FPGA-based processors (SPs) in a $2D$ grid (a board).
- Data links among nearest neighbours
- One I/O processor per board (IOP)
- A standard PC (Janus host) for each 2 boards

- We group 4×4 FPGA-based processors (SPs) in a $2D$ grid (a board).
- Data links among nearest neighbours
- One I/O processor per board (IOP)
- A standard PC (Janus host) for each 2 boards
- A total of 16 boards (256 SPs) in a Janus rack

Implementation

- The FPGAs have several small RAM blocks. We need
  - L^3 spins
  - $3L^3$ couplings (L^3 for each direction)

Implementation

- The FPGAs have several small RAM blocks. We need
  - L^3 spins
  - $3L^3$ couplings (L^3 for each direction)
  - All these items have to be moved
    to the update cells each clock cycle

Implementation

- The FPGAs have several small RAM blocks. We need
  - L^3 spins
  - $3L^3$ couplings (L^3 for each direction)
  - All these items have to be moved
    to the update cells each clock cycle
- The SP is divided in Update Engines. Each UE
  - Needs as input: 6 nearest neighbours and 6 couplings.

Implementation

- The FPGAs have several small RAM blocks. We need
  - L^3 spins
  - $3L^3$ couplings (L^3 for each direction)
  - All these items have to be moved to the update cells each clock cycle
- The SP is divided in Update Engines. Each UE
  - Needs as input: 6 nearest neighbours and 6 couplings.
  - Computes the local contribution to the energy U

Implementation

- The FPGAs have several small RAM blocks. We need
  - L^3 spins
  - $3L^3$ couplings (L^3 for each direction)
  - All these items have to be moved to the update cells each clock cycle
- The SP is divided in Update Engines. Each UE
  - Needs as input: 6 nearest neighbours and 6 couplings.
  - Computes the local contribution to the energy U
  - Addresses a pre-computed update probability table according to U (only 7 possible values)

Implementation

- The FPGAs have several small RAM blocks. We need
  - L^3 spins
  - $3L^3$ couplings (L^3 for each direction)
  - All these items have to be moved to the update cells each clock cycle
- The SP is divided in Update Engines. Each UE
  - Needs as input: 6 nearest neighbours and 6 couplings.
  - Computes the local contribution to the energy U
  - Addresses a pre-computed update probability table according to U (only 7 possible values)
  - Compares with a random number

Implementation

- The FPGAs have several small RAM blocks. We need
  - L^3 spins
  - $3L^3$ couplings (L^3 for each direction)
  - All these items have to be moved
    to the update cells each clock cycle
- The SP is divided in Update Engines. Each UE
  - Needs as input: 6 nearest neighbours and 6 couplings.
  - Computes the local contribution to the energy U
  - Addresses a pre-computed update probability table according to U (only 7 possible values)
  - Compares with a random number
  - Sets the new value of the spin.

Implementation

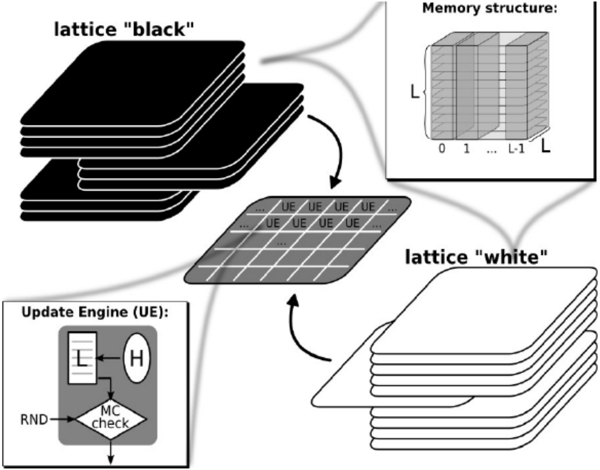
- The FPGAs have several small RAM blocks. We need
  - L^3 spins
  - $3L^3$ couplings (L^3 for each direction)
  - All these items have to be moved to the update cells each clock cycle
- The SP is divided in Update Engines. Each UE
  - Needs as input: 6 nearest neighbours and 6 couplings.
  - Computes the local contribution to the energy U
  - Addresses a pre-computed update probability table according to U (only 7 possible values)
  - Compares with a random number
  - Sets the new value of the spin.
- Pipelineable to one spin update per clock cycle per UE.

Implementation

- The FPGAs have several small RAM blocks. We need
  - L^3 spins
  - $3L^3$ couplings (L^3 for each direction)
  - All these items have to be moved
    to the update cells each clock cycle
- The SP is divided in Update Engines. Each UE
  - Needs as input: 6 nearest neighbours and 6 couplings.
  - Computes the local contribution to the energy U
  - Addresses a pre-computed update probability table
    according to U (only 7 possible values)
  - Compares with a random number
  - Sets the new value of the spin.
- Pipelineable to one spin update per clock cycle per UE.
- Each SP has enough memory for systems of linear size $L = 88$.

Implementation, general picture

An overview



Random numbers (I)

- Random number generation is a clear potential bottleneck.
- We need one RN per update and we want to do many updates per clock cycle.

Random numbers (I)

- Random number generation is a clear potential bottleneck.
- We need one RN per update and we want to do many updates per clock cycle.
- We use a shift register method.
- We have a wheel I with 62 numbers and want to generate a RN R

$$I(k) = I(k-24) + I(k-55), \qquad R = I(k) \otimes I(k-61)$$

Random numbers (I)

- Random number generation is a clear potential bottleneck.
- We need one RN per update and we want to do many updates per clock cycle.
- We use a shift register method.
- We have a wheel I with 62 numbers and want to generate a RN R

$$I(k) = I(k-24) + I(k-55), \qquad R = I(k) \otimes I(k-61)$$

- For each RN: sum two and XOR with a third.
- The wheel is then shifted,
  the computed sum filling the empty position.

Random numbers (I)

- Random number generation is a clear potential bottleneck.
- We need one RN per update and we want to do many updates per clock cycle.
- We use a shift register method.
- We have a wheel I with 62 numbers and want to generate a RN R

$$I(k) = I(k-24) + I(k-55), \qquad R = I(k) \otimes I(k-61)$$

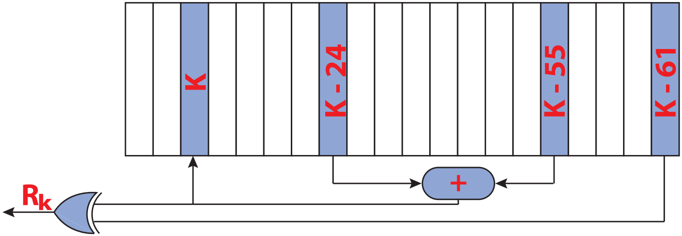
- For each RN: sum two and XOR with a third.
- The wheel is then shifted,
  the computed sum filling the empty position.
- A straightforward implementation produces a RN per step (for each wheel that we maintain).
- We need more
- Solution: implement it through logic (not memory) blocks.

Random numbers (II)

- Write the wheel in cascade-structured combinatorial logic:

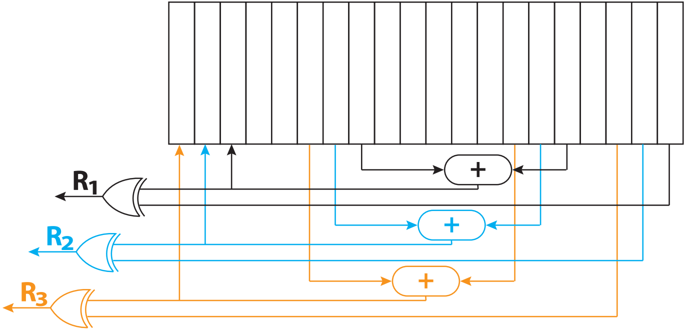
Random numbers (II)

- Write the wheel in cascade-structured combinatorial logic:
- Generation of a single RN:



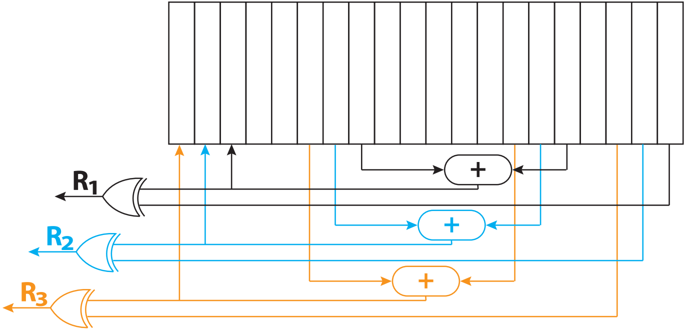
Random numbers (II)

- Write the wheel in cascade-structured combinatorial logic:
- Three RN with the same wheel in one step:



Random numbers (II)

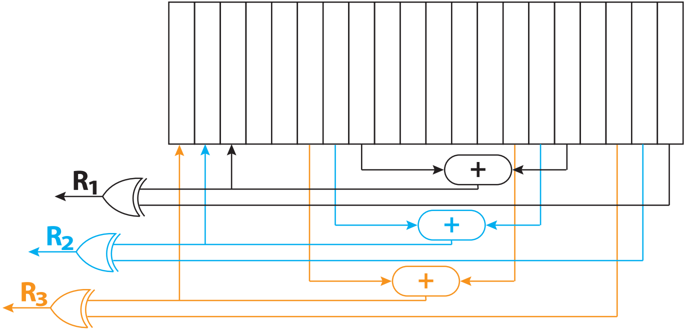
- Write the wheel in cascade-structured combinatorial logic:
- Three RN with the same wheel in one step:



- We generate 96 RN per clock cycle for each wheel.

Random numbers (II)

- Write the wheel in cascade-structured combinatorial logic:
- Three RN with the same wheel in one step:



- We generate 96 RN per clock cycle for each wheel.
- We still need to keep several wheels at the same time.

Performance

- Our FPGAs (already a few years old) are Xilinx Virtex4-LX200.
- We exhaust 88% of the available logic resources.

Performance

- Our FPGAs (already a few years old) are Xilinx Virtex4-LX200.
- We exhaust 88% of the available logic resources.
- 1024 UE on each SP

Performance

- Our FPGAs (already a few years old) are Xilinx Virtex4-LX200.
- We exhaust 88% of the available logic resources.
- 1024 UE on each SP
- The clock cycle is 62.5 MHz (one cycle each 16 ns)

$$\frac{16 \text{ ns}}{1 \text{ cycle}} \times \frac{1 \text{ cycle}}{1024 \text{ spins}} \approx 16 \text{ ps/spin}$$

Performance

- Our FPGAs (already a few years old) are Xilinx Virtex4-LX200.
- We exhaust 88% of the available logic resources.
- 1024 UE on each SP
- The clock cycle is 62.5 MHz (one cycle each 16 ns)

$$\frac{16 \text{ ns}}{1 \text{ cycle}} \times \frac{1 \text{ cycle}}{1024 \text{ spins}} \approx 16 \text{ ps/spin}$$

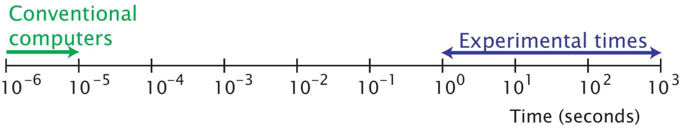
- In other words: each of the 256 SPs can take 10^{11} MCS of an $L = 80$ system in less than a month.

Performance

- Our FPGAs (already a few years old) are Xilinx Virtex4-LX200.
- We exhaust 88% of the available logic resources.
- 1024 UE on each SP
- The clock cycle is 62.5 MHz (one cycle each 16 ns)

$$\frac{16\text{ ns}}{1\text{ cycle}} \times \frac{1\text{ cycle}}{1024\text{ spins}} \approx 16\text{ ps/spin}$$

- In other words: each of the 256 SPs can take 10^{11} MCS of an $L = 80$ system in less than a month.
- In physical terms:

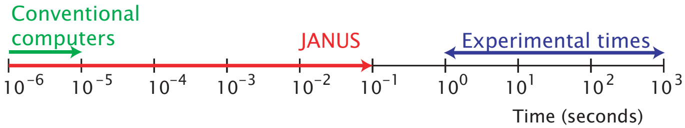


Performance

- Our FPGAs (already a few years old) are Xilinx Virtex4-LX200.
- We exhaust 88% of the available logic resources.
- 1024 UE on each SP
- The clock cycle is 62.5 MHz (one cycle each 16 ns)

$$\frac{16\ \text{ns}}{1\ \text{cycle}} \times \frac{1\ \text{cycle}}{1024\ \text{spins}} \approx 16\ \text{ps/spin}$$

- In other words: each of the 256 SPs can take 10^{11} MCS of an $L = 80$ system in less than a month.
- In physical terms:



Esquema

The spin correlation function

- The spin correlation function measures the memory at time $t + t_\mathrm{w}$ of the configuration at t_w:

$$C(t, t_\mathrm{w}) = \overline{L^{-3} \sum_{\boldsymbol{x}} \sigma_{\boldsymbol{x}}^{t+t_\mathrm{w}} \sigma_{\boldsymbol{x}}^{t_\mathrm{w}}} \quad \Longrightarrow \quad \begin{cases} C = 1 & \longrightarrow \text{ same config.} \\ C = 0 & \longrightarrow \text{ no memory.} \end{cases}$$

The spin correlation function

- The spin correlation function measures the memory at time $t + t_\mathrm{w}$ of the configuration at t_w:

$$C(t, t_\mathrm{w}) = \overline{L^{-3} \sum_{\boldsymbol{x}} \sigma_{\boldsymbol{x}}^{t+t_\mathrm{w}} \sigma_{\boldsymbol{x}}^{t_\mathrm{w}}} \quad \Longrightarrow \quad \begin{cases} C = 1 & \longrightarrow \text{ same config.} \\ C = 0 & \longrightarrow \text{ no memory.} \end{cases}$$

- In an off-equilibrium setting:

$$\underbrace{\lim_{t_\mathrm{w}\to\infty} \lim_{t\to\infty} C(t, t_\mathrm{w})}_{= 0} \neq \underbrace{\lim_{t\to\infty} \lim_{t_\mathrm{w}\to\infty} C(t, t_\mathrm{w})}_{= q_\mathrm{EA}}$$

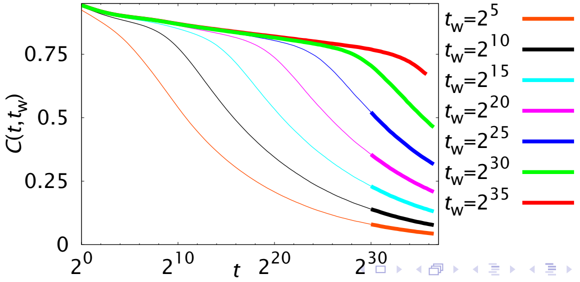
The spin correlation function

- The spin correlation function measures the memory at time $t + t_{\mathrm{w}}$ of the configuration at t_{w}:

$$C(t, t_{\mathrm{w}}) = \overline{L^{-3} \sum_{\boldsymbol{x}} \sigma_{\boldsymbol{x}}^{t+t_{\mathrm{w}}} \sigma_{\boldsymbol{x}}^{t_{\mathrm{w}}}} \quad \Longrightarrow \quad \begin{cases} C = 1 & \longrightarrow \text{ same config.} \\ C = 0 & \longrightarrow \text{ no memory.} \end{cases}$$

- In an off-equilibrium setting:

$$0 = \lim_{t_{\mathrm{w}}\to\infty} \lim_{t\to\infty} C(t, t_{\mathrm{w}}) \neq \lim_{t\to\infty} \lim_{t_{\mathrm{w}}\to\infty} C(t, t_{\mathrm{w}}) = q_{\mathrm{EA}}$$

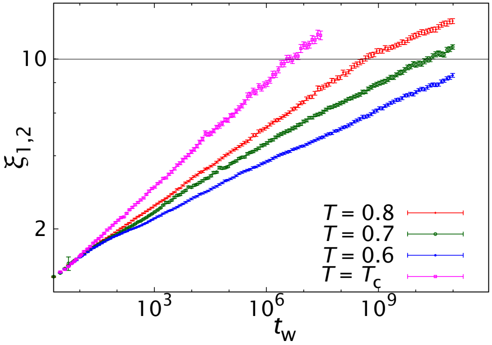


The coherence length

- Slow growth of coherent domains.
- We measure the coherence length ξ and fit to $\xi = A(T)\,t_{\mathrm{w}}^{1/z(T)}$.

The coherence length

- Slow growth of coherent domains.
- We measure the coherence length ξ and fit to $\xi = A(T)\, t_w^{1/z(T)}$.

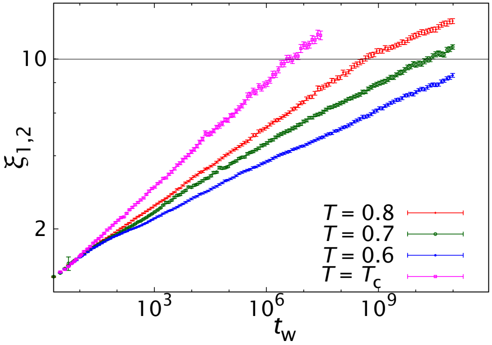


$$
\begin{aligned}
z(T_c) &= 6.86(16) \\
z(0.73\,T_c) &= 9.42(15) \\
z(0.64\,T_c) &= 11.8(2) \\
z(0.55\,T_c) &= 14.1(3)
\end{aligned}
$$

$$z(T) \simeq z(T_c)\,T_c/T$$

The coherence length

- Slow growth of coherent domains.
- We measure the coherence length ξ and fit to $\xi = A(T)\, t_\mathrm{w}^{1/z(T)}$.



$$z(T_\mathrm{c}) = 6.86(16)$$
$$z(0.73\,T_\mathrm{c}) = 9.42(15)$$
$$z(0.64\,T_\mathrm{c}) = 11.8(2)$$
$$z(0.55\,T_\mathrm{c}) = 14.1(3)$$

$$z(T) \simeq z(T_\mathrm{c})\, T_\mathrm{c}/T$$

- For $T \geq 0.64\,T_\mathrm{c}$ we begin to see finite-size effects, even with $L = 80$!

Summary of Janus' physics work

Non-equilibrium spin-glass dynamics

- We have followed the dynamics from picoseconds to 0.1 s.
- We find evidence for non-coarsening dynamics.
- PRL **101**, 157201 (2008), JSP **135** 1121 (2008).

Summary of Janus' physics work

Non-equilibrium spin-glass dynamics

- We have followed the dynamics from picoseconds to 0.1 s.
- We find evidence for non-coarsening dynamics.
- PRL **101**, 157201 (2008), JSP **135** 1121 (2008).

The equilibrium spin-glass phase

- We have studied the low-temperature spin-glass phase
- Parallel tempering with Janus
- Equilibrate 10^3 up to $L = 32$ and down to $T = 0.64T_c$.
- A total of over 10^{21} spin updates
- We find evidence in favour of the RSB picture,
  at least for experimentally relevant scales.
- PRL **105**, 177202 (2010), JSTAT **(2010)** P06026.

Summary of Janus' physics work

Critical behaviour of the Potts glass

- We have studied the three-dimensional Potts glass with $p = 4, 5, 6$.
- We find clear spin-glass transitions, but no ferromagnetic transition.
- PRB, **79**, 184408 (2009), JSTAT **(2010)** P05002.

Summary of Janus' physics work

Critical behaviour of the Potts glass

- We have studied the three-dimensional Potts glass with $p = 4, 5, 6$.
- We find clear spin-glass transitions, but no ferromagnetic transition.
- PRB, **79**, 184408 (2009), JSTAT **(2010)** P05002.

Work in progress

- We are currently studying the dynamics and critical point of the Edwards-Anderson spin-glass in $D = 3, 4$, with an applied magnetic field.

Esquema

Limitations of Janus

- We have only studied isothermal aging.

Limitations of Janus

- We have only studied isothermal aging.
- Close to T_c the coherence length grows relatively quickly and we begin to see finite-size effects with $L = 80$

Limitations of Janus

- We have only studied isothermal aging.
- Close to T_c the coherence length grows relatively quickly and we begin to see finite-size effects with $L = 80$
- We need larger sizes for more complicated protocols, with varying temperature

Limitations of Janus

- We have only studied isothermal aging.
- Close to T_c the coherence length grows relatively quickly and we begin to see finite-size effects with $L = 80$
- We need larger sizes for more complicated protocols, with varying temperature
- Each SP can only hold up to $L = 88$, but we can spread the system accross several in the same board.
- This way, we are curently testing a code capable of taking 10^{11} MCS on an $L = 256$ lattice in under a month.

Limitations of Janus

- We have only studied isothermal aging.
- Close to T_c the coherence length grows relatively quickly and we begin to see finite-size effects with $L = 80$
- We need larger sizes for more complicated protocols, with varying temperature
- Each SP can only hold up to $L = 88$, but we can spread the system accross several in the same board.
- This way, we are curently testing a code capable of taking 10^{11} MCS on an $L = 256$ lattice in under a month.
- Still, with Janus, we are at the threshold of actual experimental scales.

Janus II

- We already have full funding for the next generation: Janus II.
- Will begin construction in a few months

Janus II

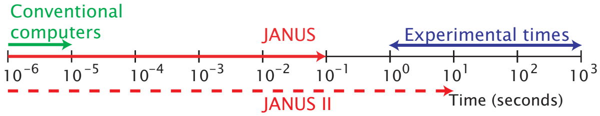
- We already have full funding for the next generation: Janus II.
- Will begin construction in a few months
- Innovations
  - Faster FPGAs.
  - Connections between SPs on different boards (a toroidal net of $4 \times 4 \times 16$ SPs).
  - Some additional RAM

Janus II

- We already have full funding for the next generation: Janus II.
- Will begin construction in a few months
- Innovations
  - Faster FPGAs.
  - Connections between SPs on different boards (a toroidal net of $4 \times 4 \times 16$ SPs).
  - Some additional RAM
- We want a more general machine, to explore other potential applications (Molecular Dynamics, etc.)

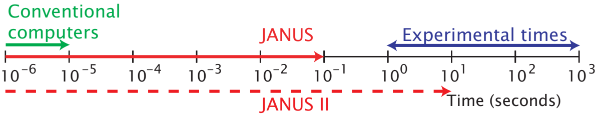
Janus II

- We already have full funding for the next generation: Janus II.
- Will begin construction in a few months
- Innovations
  - Faster FPGAs.
  - Connections between SPs on different boards (a toroidal net of $4 \times 4 \times 16$ SPs).
  - Some additional RAM
- We want a more general machine, to explore other potential applications (Molecular Dynamics, etc.)
- For spin glasses, capable of simulating an $L = 1000$ system (as large as experimental samples).



Janus II

- We already have full funding for the next generation: Janus II.
- Will begin construction in a few months
- Innovations
  - Faster FPGAs.
  - Connections between SPs on different boards (a toroidal net of $4 \times 4 \times 16$ SPs).
  - Some additional RAM
- We want a more general machine, to explore other potential applications (Molecular Dynamics, etc.)
- For spin glasses, capable of simulating an $L = 1000$ system (as large as experimental samples).



- Open to applications from external groups

Janus III

- We are already thinking about Janus III

Janus III

- We are already thinking about Janus III
- The architecture is still undecided (GPUs, FPGAs, mixed?)
    - Will depend on the applications

Janus III

- We are already thinking about Janus III
- The architecture is still undecided (GPUs, FPGAs, mixed?)
  - Will depend on the applications
  - Will take advantage of our experience with Janus II (already planned as an intermediate step)

Janus III

- We are already thinking about Janus III
- The architecture is still undecided (GPUs, FPGAs, mixed?)
    - Will depend on the applications
    - Will take advantage of our experience with Janus II (already planned as an intermediate step)
- Open to new collaborators, even for the design stage

Conclusions

- We have presented Janus, a special-purpose computer for high-performance scientific computing

Conclusions

- We have presented Janus, a special-purpose computer for high-performance scientific computing
- Janus' FPGA architecture permits a high level of parallelism.

Conclusions

- We have presented Janus, a special-purpose computer for high-performance scientific computing
- Janus' FPGA architecture permits a high level of parallelism.
- Extremely efficient for non-equilibrium dynamics or low-temperature equilibrium, where very long simulations for each sample are needed.

Conclusions

- We have presented Janus, a special-purpose computer for high-performance scientific computing
- Janus' FPGA architecture permits a high level of parallelism.
- Extremely efficient for non-equilibrium dynamics or low-temperature equilibrium, where very long simulations for each sample are needed.
- Also very efficient for critical-point studies, where the simulations are shorter, but cheaper alternatives (GPUs) are competitive there.

Conclusions

- We have presented Janus, a special-purpose computer for high-performance scientific computing
- Janus' FPGA architecture permits a high level of parallelism.
- Extremely efficient for non-equilibrium dynamics or low-temperature equilibrium, where very long simulations for each sample are needed.
- Also very efficient for critical-point studies, where the simulations are shorter, but cheaper alternatives (GPUs) are competitive there.
- Janus is very energy-efficient: the whole rack needs only ≈ 11 kW and is capable of ≈ 8.75 Gops/W.