Janus: FPGA Based System for Scientific Computing

Filippo Mantovani
Physics Department
Università degli Studi di Ferrara

Ferrara, 28/09/2009
Overview:

1. The physical problem:
   - Ising model and Spin Glass in short;

2. The computational challenge:
   - Monte Carlo “tools”;
   - Computational features of Spin Glass problems;

3. The Janus system:
   - an exotic attempt to simulate spin systems;

4. Conclusions
Statistical mechanics tries to describe the macroscopic behavior of matter in terms of average values of microscopic structure.

For instance, try to explain why magnets have a transition temperature (beyond which they lose their magnetic state)...
The Ising model

Each spin takes only the values +1 (up) or -1 (down) and interacts only with its nearest neighbours in a discrete D-dimensional mesh (e.g. 2D).

Energy of the system for a given configuration \( \{ S_i \} \):  
\[
U(\{ S_i \}) = -J \sum_{\langle ij \rangle} s_i s_j
\]

Note that:

A) Each configuration \( \{ S_i \} \) has a probability given by the Boltzmann factor:  
\[
P(\{ S_i \}) \propto e^{-U(\{ S_i \})/kT} \equiv e^{-\beta U(\{ S_i \})}
\]

B) Macroscopic observables have averages obtained by summing on each configurations, each weighted by its probability.

Magnetization:  
\[
M \equiv \sum_i s_i \quad \langle M \rangle \propto \sum_{\{ S_i \}} M(\{ S_i \}) P(\{ S_i \})
\]
The Ising model

What about the “J”?!?
In the Ising model we assume that $J$ are constant ($=+1$ or $-1$). The energy profile is therefore smooth (and in this case symmetric):

$$U(\{S_i\}) = -J \sum_{\langle ij \rangle} S_i S_j$$

![Diagram showing energy profile at different temperatures](image)

- $T > T_c$
- $T = T_c$
- $T < T_c$

![Diagram showing magnetization vs. temperature](image)

- $T_c$
The Spin Glass model

Spin glasses are a generalization of Ising systems. An apparently trivial change in the energy function:

$$ U(\{ S_i \}) = -\sum_{ij} J_{ij} s_i s_j , \quad s = \{+1, -1\}, \quad J = \{+1, -1\} $$

makes spin glass models much more interesting and complex than Ising systems: if we leave J NOT fixed, we obtain 2 important consequences:

1) frustration:

2) corrugation in energy function:

---

Filippo Mantovani, 28/09/2009
Problems around that...

Problems like finding the minimum energy configuration at a given temperature or finding the critical temperature are absolutely not trivial and represent a computational nightmare!!

**HINT:**
If we consider a 3D lattice, with 48 spins per side, we obtain that the possible configurations are $2^{48^3}$!!!
Overview:

1. The physical problem:
   - Spin Glass and Ising model in short;

2. The computational challenge:
   - Monte Carlo “tools”;
   - Computational features of Spin Glass problems;

3. The Janus system:
   - an exotic attempt to simulate spin systems;

4. Conclusions
Monte Carlo algorithms

As seen before, relative small lattices produce a huge number of possible configurations.

Monte Carlo algorithms are tools allowing us to visit configuration according to its probability.

The Markov chains theory assures us that is valid the **property of ergodicity**:

\[ \pi_i = \lim_{t \to \infty} p_i(t) \]

So the probability of the configuration \( i \), after a time long enough, tends to a probability \( \pi_i \) (**equilibrium condition**).
Monte Carlo algorithms

A direct consequence of this Markov chains property is that we can produce configurations that are distributed following a given distribution.

\[ C_{i}^{MC} ; \quad i = 1 \ldots n \]

The “good news” is that observables are trivially computed as \textbf{unweighted sums} of Monte Carlo generated configurations*:

\[ \langle f \rangle \propto \sum_{\{C\}} f(C) P(C) = \sum_{i} f(C_{i}^{MC}) \]

*we do not need any more to sum the observable weighted with the probability.
The Metropolis algorithm

Assuming a 3D lattice composed of spins, we can update each lattice site following these steps:

✔ consider one spin \( s_0 \) and compute his local energy \( E = \sum_{\langle 0j \rangle} J_{ij} s_j s_0 \)

✔ flip the value of the spin: \( s_0' = -s_0 \)

✔ compute the new local energy \( E' = \sum_{\langle 0j \rangle} J_{ij} s_j s_0' \)

✔ if \( E' < E \) then the spin flip is accepted; else we compute \( X = \exp\left(-\frac{E' - E}{T}\right) \) and generate a pseudo-random number \( R \) (\( 0 \leq R \leq 1 \)):
  - if \( R < X \) then the spin flip is accepted as well
  - else remains \( s \).

**NOTE:** If we replace the spin lattice with the topology of a random graph the algorithm keeps working (e.g. this is exactly what we used to implement graph coloring).
for ( k=0 ; k<SIDE; k++ ) {
    for ( j=0 ; j<SIDE; j++ ) {
        for ( i=0 ; i<SIDE; i++ ) {
            // calculating the local energy
            LE = spin [(i+1)%SIDE][j][k] * Jx[(i+1)%SIDE][j][k] + 
                 spin [(i-1)%SIDE][j][k] * Jx[(i-1)%SIDE][j][k] + 
                 spin [i][(j+1)%SIDE][k] * Jy[i][(j+1)%SIDE][k] + 
                 spin [i][(j-1)%SIDE][k] * Jy[i][(j-1)%SIDE][k] + 
                 spin [i][j][(k+1)%SIDE] * Jz[i][j][(k+1)%SIDE] + 
                 spin [i][j][(k-1)%SIDE] * Jz[i][j][(k-1)%SIDE] ;

            // calculating the energy variation after the spin flip spin[i][j][k]
            deltaE = -2 * spin[i][j][k] * LE ;

            // if energy decrease we accept the spin flip...
            if ( deltaE < 0 ) {
                spin [i][j][k] = - spin[i][j][k] ;
            } else {
                // ...else we pick a pseudo-random number and compare it
                // with a given function of deltaE and T (temperature)
                // in order to accept the spin flip
                if ( rand() < exp( - deltaE / T ) ) {
                    spin [i][j][k] = - spin[i][j][k] ;
                }
            }
        }
    }
}
Computational features

- Operations on spins $\equiv$ bit-manipulation.
- Computation of a limited set of exponential. (We can store them in LUTs).
- Random numbers should be of “good quality” and with a long period.
- A huge degree of available parallelism.
- Regular program flow. (orderly loops on the grid sites)
- Regular, predictable memory access pattern.
- Information-exchange (processor $\leftrightarrow$ memory) is huge, however the size of the data-base is tiny. (On-chip memory seems to be ideal...)
The PC approach...

Using standard architectures as Spin Glass engines two different algorithms are commonly used:

**Synchronous Multi-Spin Coding (SMSC):**
- one CPU handles one single system;
- replicas are handled on a farm of several CPU (128 - 256);
- at every step each CPU update in parallel N spins (N=2 or 4);

**NOTE:** bottleneck is the number of floating point random numbers can be generated in parallel.

**Asynchronous Multi-Spin Coding (AMSC):**
- each CPU handles several (64 - 128) systems in parallel;
- replicas are handled on CPU farm (smaller than previous);
- at every step each CPU update in parallel 64 - 128 spins of different systems;
- on each single CPU random number is shared among all systems.
Overview:

1. The physical problem:
   - Spin Glass and Ising model in short;

2. The computational challenge:
   - Monte Carlo “tools”;
   - Computational features of Spin Glass problems;

3. The Janus system:
   - an exotic attempt to simulate spin systems;

4. Conclusions
The Janus project

A collaboration of:

- Universities of Rome “La Sapienza” and Ferrara
- Universities of Madrid, Zaragoza, Extremadura
- BIFI (Zaragoza)
- Eurotech
The Janus approach

Preamble: We split the 3D lattice in 2 parts, “black” spins and “white” spins (slice = checkerboard). We cannot update in the same time black spins and white spins.

✔ We store the lattice within the embedded memories of the FPGA.

✔ We use a set of registers to have no latency access to a “small set” of spins (and couplings)

✔ We house a set (as large as possible) of Update Engines...
The Janus approach

Each Update Engine:
- computes the local contribution to $U$:
  \[ U = - \sum_{ij} s_i J_{ij} s_j \]
- addresses a probability table according with $U$;
- compares with a freshly generated random number;
- sets the new spin value;
The Janus approach

A whole picture:

lattice "black"

Memory structure:

lattice "white"

Update Engine (UE):

- L
- H
- RND
- MC check
The toy...

The basic hardware elements are:

- a 2-D grid of 4 x 4 FPGA-based processors (SPs);
- **data links** among nearest neighbours on the FPGA grid;
- one control processor on each board (IOP) with 2 gbit-ethernet ports;
- a standard PC (**Janus host**) for each 2 Janus boards.
Janus summary:

- 256 Xilinx Virtex4-LX200 (+16 IOPs);
- 1024 update engines on each processor;
- pipelineable to one spin update per clock cycle;
- 88% of available logic resources;
- using a bandwidth of \( \sim 12000 \) read bits + \( \sim 1000 \) written bits per clock cycle \( \Rightarrow 47\% \) of available on-chip memory;
- system clock at 62.5 MHz \( \Rightarrow 16 \text{ ps average spin update time.} \)
- 1 SP \( \sim 40\text{W}, \) 16 SP \( \sim 640\text{W}, \) rack \( \sim 11\text{KW} \)
The relevant performance measure is the average update time per spin $(R)$ measured in pico-sec per spin.

For each FPGA in the system we have:

$$R = 1024 \text{ spin updates for each clock cycle}$$

$$= 1024 \times 16 \text{ ns}$$

$$\sim 16 \text{ ps / spin}$$

For one complete 16 processors board:

$$R = \frac{1}{16} \times 1024 \text{ spin updates for each clock cycle}$$

$$= \frac{1}{16} \times 1024 \times 16 \text{ ns}$$

$$\sim 1 \text{ ps / spin}$$
Performance
(for computer scientists)

Each update-core performs 11 + 2 sustained pipelined operations* per clock cycle (@ 62.5 MHz)
1024 update engines perform:

\[
\begin{align*}
1024 \times 13 \text{ ops} / 16 \text{ ns} &= 832 \text{ Gops} \\
1024 \times 3 \text{ ops} / 16 \text{ ns} &= 192 \text{ Gops}^**
\end{align*}
\]

* these operations are on very short data words!!!
** submitted to Gordon Bell Prize 2008.

Sustaining these performance requires a huge bandwidth to/from (fine-grained) memory:

- processing just one bit implies getting 12 more bits from memory: 6 neighbours + 6 “J” = 12 bits;
- generating pseudo-randoms requires 3 32-bit data;
- 1 more 32-bit data (from the LUT) is necessary for MC step;

All in all the memory bandwidth sustained is:

\[
140 \text{ bit} \times \frac{1024}{16} \text{ ns} \sim 1 \text{ TB/s}
\]
## Performance

### Comparing with more conventional systems...

Real example: $10^{12}$ Monte Carlo steps, $L = 64$.

<table>
<thead>
<tr>
<th></th>
<th>Janus</th>
<th>AMSC</th>
<th>SMSC</th>
<th>CBE</th>
</tr>
</thead>
<tbody>
<tr>
<td>processor statistic</td>
<td>1 SP</td>
<td>1 CPU</td>
<td>1 CPU</td>
<td>2 CPU</td>
</tr>
<tr>
<td>wall-clock time energy</td>
<td>1 (16)</td>
<td>1 (128)</td>
<td>1 (4)</td>
<td>1 (4)</td>
</tr>
<tr>
<td></td>
<td>50 days</td>
<td>770 years</td>
<td>25 years</td>
<td>15 months</td>
</tr>
<tr>
<td></td>
<td>2,7 GJ</td>
<td>2,3 TJ</td>
<td>78,8 GJ</td>
<td>7.7 GJ</td>
</tr>
</tbody>
</table>

|                          | 256 SPs     | 2 CPUs      | 256 CPUs    | 128 CPUs    |
|                          | 256         | 256         | 256         | 256         |
|                          | 50 days     | 770 years   | 25 years    | 15 months   |
|                          | 43 GJ       | 4,6 TJ      | 20 TJ       | 500 GJ      |
Power consumption

**Power:**

1 FPGA ........................................... ~40W
1 Janus system ............................... ~11KW
1 domestic power supply.... ~5KW

**Operations per Watts:**

Janus .......................................... ~8.75 Gops/W
Best case Green500* ......... ~0.536 Gflops/W

A run of 25 days** uses 15 oil barrels; to obtain the same results with a cluster of PCs we need 12000 oil barrels***.

* BladeCenter QS22 Cluster, PowerXCell 8i 4.0 Ghz, University of Warsaw (June 2009).
** Run period of the first intensive Janus simulation (April 2008).
*** We suppose an unreal condition of scalability.
Janus gallery: SP
Janus gallery: IOP
Janus gallery: board
Janus gallery: rack
Overview:

1. The physical problem:
   - Spin Glass and Ising model in short;
   - LQCD vs statistical mechanics;

2. The computational challenge:
   - Monte Carlo "tools";
   - Computational features of Spin Glass problems;

3. The Janus system:
   - an exotic attempt to simulate spin systems;

4. Conclusions
Conclusions

✔ the Janus system is an heterogeneous massively parallel systems;

✔ made by highly parallel processors implemented on FPGA;

✔ delivers impressive performance: 1 SP is equivalent to ~100 PC and ~30-50 new multi-core architectures;

✔ used for scientific applications like Spin Glass, but in principle available for any kind of problems fitting the topology of the machine and the resources of the FPGAs.

✔ uses NON-challenge technology... (plug the LEGO bricks and play)
A slightly broader view...

The study of these systems:
- for a **physicist** means a better understanding of condensed matter;
- for a **computer scientist** means developing new algorithms and improving old ones;
- for a “**computer mason**” (like me) means developing a massively parallel high performance system.

Moreover:
All these systems are surprisingly strong related with other areas, like e.g. graph coloring.
About the random numbers

The pseudo random numbers are generated using a shift-register of N words of 32 bits and can generate up to $\sim 100$ random numbers per clock cycle.

**NOTE:** 1 shift-register generating 32 numbers uses $\sim 2\%$ of the FPGA resources ($\sim 2500$ Slices).