Solving combinatorial problems on large multiGPU clusters: breaking the challenge of the Langford problem

Julien Loiseau
Christophe Jaillet
François Alin
Michaël Krajecki

CReSTIC - ROMEO
Université de Reims Champagne-Ardenne

Journée ROMEO, 11 juin 2015
1. Combinatorial problems

2. Massively parallel architectures

3. Miller’s method = backtrack algorithm

4. Godfrey’s method

5. Conclusions
Plan

1. Combinatorial problems
   - Situation in Operational Research
   - Langford problem as a case study

2. Massively parallel architectures

3. Miller’s method = backtrack algorithm

4. Godfrey’s method

5. Conclusions
combinatorial problem
- discrete problem
- combinatorial explosion
- combinatorial optimization / combinatorial search

? satisfiable
- build one or all the solutions
- determine the number of solutions

CSP = Constraint Satisfaction Problem
- combinatorial problem $\Rightarrow$ NP-Complete
- NP-Complete $\Rightarrow$ SAT/CSP
- tree representation
Combinatorial problems

Langford problem

\( L(2, 3) = 1 \)

numeric representation

\( n = 4k \)
\( n = 4k - 1 \)

\( L(2, 19) \) in 1999
- sequential: 2 years \( \frac{1}{2} \)
- distributed: 2 months

\( L(2, 20) \) in 2002
- algebraic method

<table>
<thead>
<tr>
<th>( n )</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>26</td>
</tr>
<tr>
<td>8</td>
<td>150</td>
</tr>
<tr>
<td>11</td>
<td>17792</td>
</tr>
<tr>
<td>12</td>
<td>108144</td>
</tr>
<tr>
<td>15</td>
<td>39809640</td>
</tr>
<tr>
<td>16</td>
<td>326721800</td>
</tr>
<tr>
<td>19</td>
<td>256814891280</td>
</tr>
<tr>
<td>20</td>
<td>2636337861200</td>
</tr>
<tr>
<td>23</td>
<td>3799455942515480</td>
</tr>
<tr>
<td>24</td>
<td>46845158056515900</td>
</tr>
<tr>
<td>27</td>
<td>??</td>
</tr>
</tbody>
</table>
Plan

1. Combinatorial problems

2. Massively parallel architectures
   - Cluster architecture
   - GPU

3. Miller’s method = backtrack algorithm

4. Godfrey’s method

5. Conclusions
Massively parallel architectures

parallelism

- nodes ⇔ interconnect
- machines
- 3 levels of parallelism

ROMEO

- Fat-tree with InfiniBand
  - CPU : E5-2650v2 2.6GHz, 8c
  - GPU : NVIDIA K20Xm
  → TOP500 and GREEN500

URCA Solving combinatorial problem on multiGPU cluster
Massively parallel architectures

GPU

- SIMD/SIMT

- 1000+ elementary processors
  - specific processors
  - simplified
  - synchronization

NVIDIA/CUDA

- hierarchical memory
- threads, blocks and grid
- warps : 32 threads

Divergence

- SIMT, synchronization

- avoid desynchronization
void saxpy(int n, float a, float *x, float *y)
{
    for(int i = 0 ; i < n ; i++)
    {
        y[i] = a*x[i] + y[i] ;
    }
}
...

int N = 1<<20 ;
saxpy(N, 2.3, x, y) ;

__global__
void saxpy(int n, float a, float *x, float *y)
{
    int i = blockIdx.x*blockDim.x + threadIdx.x ;
    if(i < n) y[i] = a*x[i] + y[i] ;
}
...

int N = 1<<20 ;
cudaMemcpy(d_x, x, N, cudaMemcpyHostToDevice) ;
saxpy<<<4096,256>>>(N, 2.3, d_x, d_y) ;
cudaMemcpy(y, d_y, N, cudaMemcpyDeviceToHost) ;
Plan

1. Combinatorial problems
2. Massively parallel architectures
3. Miller’s method = backtrack algorithm
   - Backtrack resolution
   - Parallel resolution
   - Experimental methodology
   - Results
4. Godfrey’s method
5. Conclusions
CSP resolution

- AC / ... / backtrack/backjumping/forwardchecking/...
- variable order, choice heuristic ...

Langford problem $L(2, 3)$

- level $\Rightarrow$ pair
- position conflict
Miller’s method = backtrack algorithm

parallel resolution

**depth distribution**

- \( p \) (server) + \( q \) (client) + \( r \) (CPU/GPU) = \( n \)
- large number of tasks \( \Rightarrow \) load balancing
- server, client and CPU \( \Rightarrow \) backtrack
- GPU : backtrack or vectorized

![Diagram showing depth distribution and load balancing](image-url)
general methodology

- blocks and grid size $\Rightarrow$ registers
- streams
- CPU cores involved to feed the GPU
- distribution depths
Miller’s method = backtrack algorithm

Experimental methodology

- Factor: Backtrack, Regularized
- Threads per block: 64-96, 64-96
- Streams: 1, 3
- CPU cores for GPU: 0, 3-4
- Server depth: 3-4, 3-4
- CPU/GPU depth: 9, 5
- Tasks distribution: 80% for GPU, -

URCA
Solving combinatorial problem on multiGPU cluster
11 juin 2015 14 / 25
40 machines + 1 server:

<table>
<thead>
<tr>
<th>$n$</th>
<th>Backtrack CPU</th>
<th>Regularized CPU + GPU</th>
<th>Backtrack CPU + GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>21.219</td>
<td>14.344</td>
<td>6.637</td>
</tr>
<tr>
<td>17</td>
<td>200.306</td>
<td>120.544</td>
<td>37.166</td>
</tr>
<tr>
<td>18</td>
<td>1971.019</td>
<td>1178.261</td>
<td>408.501</td>
</tr>
<tr>
<td>19</td>
<td>22594.221</td>
<td>13960.871</td>
<td>4602.294</td>
</tr>
</tbody>
</table>

**regularized**
- $\approx$ CPU backtracking
- $\times 200\,000$ nodes
- GPU: 80% of the computation

**backtrack GPU**
- 3× faster
- GPU: 65% of the computation
258 machines + 1 server:

<table>
<thead>
<tr>
<th>$n$</th>
<th>CPU</th>
<th>CPU + GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>29.847</td>
<td>7.3</td>
</tr>
<tr>
<td>18</td>
<td>290.052</td>
<td>73.604</td>
</tr>
<tr>
<td>19</td>
<td>3197.526</td>
<td>803.524</td>
</tr>
<tr>
<td>20</td>
<td>–</td>
<td>9436.961</td>
</tr>
<tr>
<td>21</td>
<td>–</td>
<td>118512.420</td>
</tr>
</tbody>
</table>

258 nodes on ROMEO

- Miller’s method previous limits: $L(2, 19)$
- now $L(2, 20)$ and $L(2, 21)$
- 258 machines $\rightarrow$ speedup 230
Plan

1. Combinatorial problems
2. Massively parallel architectures
3. Miller’s method = backtrack algorithm
4. Godfrey’s method
   - Method
   - Optimizations
   - Implementation
   - Distribution
   - Results
5. Conclusions
algebraic method

- specific for the Langford problem
- based on cubes’ positions
- simplifications

\[ L(2, 3) \Rightarrow X = (X_1, X_2, X_3, X_4, X_5, X_6) \]

\[ F(X, 3) = (X_1X_3 + X_2X_4 + X_3X_5 + X_4X_6) \times (X_1X_4 + X_2X_5 + X_3X_6) \times (X_1X_5 + X_2X_6) = \prod_{i=1}^{n} \sum_{k=1}^{2n-i-1} x_kx_{k+i+1} \]

\[ \sum_{(x_1, \ldots, x_{2n}) \in \{-1, 1\}^{2n}} \left( \prod_{i=1}^{2n} x_i \right) \prod_{i=1}^{n} \sum_{k=1}^{2n-i-1} x_kx_{k+i+1} = 2^{2n+1} L(2, n) \]
Godfrey’s method

Optimizations

symmetry

- global sign changing \( \Rightarrow F(-X, n) = F(X, n) \)
- half sign changing \( \Rightarrow \) pair or impair variables
- symmetry summing

sum order

- change a single bit \( \Rightarrow \) use the previous sum
- Gray code sequence

\[
\begin{align*}
0 0 0 0 & \Rightarrow 0 \\
0 0 0 1 & \Rightarrow 1 \\
0 0 1 1 & \Rightarrow 3 \\
0 0 1 0 & \Rightarrow 2 \\
\ldots
\end{align*}
\]
Godfrey’s method

**Implementation**

Big integer arithmetic

- \( L(2, 16) \Rightarrow 70 \) bits
- big integer representation needed
- specific for the problem

![Diagram showing big integer representation and operations]

- in progress: assembly big integer on CPU/GPU
Experimental tuning

→ blocks and grid size
→ CPU/GPU distribution
→ distribution depth

![Graphs showing experimental tuning results]
Workflow distribution

**static distribution**
- MPI + [OpenMP/Cuda]
- one reservation

**dynamic distribution**
- Best-Effort + [OpenMP/Cuda]
- server + jobs
- requeue/cancel

Queue feeding

Task handler

32768 tasks

Finite tasks queue

Ok

!Ok

256 max active Tasks

Tasks queue

Job

Job

Job

URCA
Solving combinatorial problem on multiGPU cluster

11 juin 2015 22 / 25
$L(2, 27)$ resolution using Best-Effort on ROMEO

$\rightarrow$ 2 days of computation

$\rightarrow$ 70% of ROMEO

$\approx$ 181 machines

$L(2, 27) = 111\, 683\, 606\, 778\, 027\, 803\, 456$
Plan

1. Combinatorial problems
2. Massively parallel architectures
3. Miller’s method = backtrack algorithm
4. Godfrey’s method
5. Conclusions
Conclusions

**backtrack resolution**
- resolution methods using three levels of parallelism
- validation of the method
- Langford limit up to $L(2, 20) - L(2, 21)$
  - GPU efficiency: 80% of the computation

**Godfrey’s method**
- Langford limit up to $L(2, 27)$
- GPU: 65% of the computation

**perspectives**
- solve $L(2, 28)$
- improve the method on other problems
- optimization problems