Towards the Parallel Resolution of the Langford Problem on a Cluster of GPU Devices

Hervé Deleau*, Christophe Jaillet*, Michaël Krajecki*, Julien Loiseau*,
Luiz Angelo Steffenel* and François Alin†
*Université de Reims Champagne-Ardenne, Reims, France
†Lycée Franklin Roosvelt, Reims, France
{*firstname.lastname*}@*univ-reims.fr, julien.loiseau@etudiant.univ-reims.fr*

I. The Langford problem

The Langford problem is a classic permutation problem [1], [2]. While observing his son manipulating blocks of different colors, Langford noticed that it was possible to arrange three pairs of blocks of different colors (e.g., yellow, red, blue) in such a way that color 1 cubes were separated by 1 block, color 2 by 2 blocks, etc. (Fig. 1).

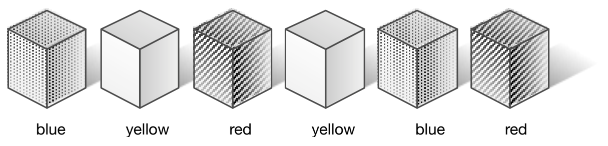

Figure 1. L(2,3): arrangement for 6 blocks of 3 colors.

The n^{th} instance of the Langford problem consists in counting the number $L(2, n)$ of such pairs arrangements (up to a symmetry). This study considers the standard Langford problem but could be generalized to any number s of blocks having the same color, in order to get the $L(s, n)$ value. Martin Gardner presented instance 4 of the problem (2 cubes and 4 colors) as being part of a collection of small mathematical games and stated that $L(2, n)$ has solutions for all n such that $n = 4k$ or $n = 4k - 1$ for $k \in \mathbb{N} \setminus \{0\}$.

A. Miller's algorithm: a tree search approach

The Langford problem can be modeled as a tree search problem where look for all possible solutions. In order to solve $L(2, n)$, we consider a tree of height n where:

- every node of the tree corresponds to the place in the sequence of the cubes of a determined color;
- at the depth p, the first node corresponds to the place of the first cube of color p in first position, and the ith node corresponds to the positioning of the first cube of color p in position i, where $i \in [1, 2n - 1 - p]$;
- every leaf of the tree symbolizes the positions of all the cubes;
- a leaf is a solution if it respects the color constraint defined by the Langford problem: all the cubes must be in different places.

This search tree is usually implemented in a bottom-up approach (Fig. 2), where the top color is the nth color, as this approach allows tree pruning to remove symmetric results and unsolvable branches.

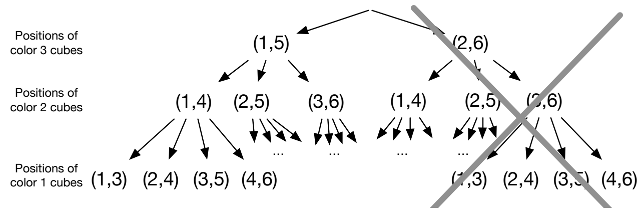

Figure 2. Search tree for $L(2, 3)$ with symmetry pruning.

B. Godfrey's algorithm: algebraic method

The Miller's approach, limited to this naive tree search evaluation with backtracking, suffers from combinatorial explosion. It allowed to get $L(2, 19)$ in 1999 after 2.5 years on a DEC alpha computer, but the compute time is estimated to be 10 times higher from an instance to the following one.

In 2002, an algebraic representation of the Langford problem has been proposed by Godfrey[1].
Consider $L(2, 3)$ and $X = (X_1, X_2, X_3, X_4, X_5, X_6)$. It proposes to model instance 3 by $F(X, 3) = (X_1X_3 + X_2X_4 + X_3X_5 + X_4X_6) \times (X_1X_4 + X_2X_5 + X_3X_6) \times (X_1X_5 + X_2X_6)$. In this approach, each term represents a position for both cubes of a given color ; the number of solutions is equal to the coefficient of $X_1X_2X_3X_4X_5X_6$ in the polynomial development. More generally, the number of solutions of instance n corresponds to the coefficient of $X_1X_2X_3X_4X_5...X_{2n}$ in $F(X, n)$.

[1]http://legacy.lclark.edu/~miller/langford/godfrey/method.html

If $G(X,n) = X_1 ... X_{2n} F(X,n)$ then Godfrey has shown that:

$$\sum_{(x_1,...x_{2n}) \in \{-1,1\}^{2n}} G(X,n)_{(x_1...x_{2n})} = 2^{2n+1} L(2,n)$$

So:

$$\sum_{(x_1...x_{2n}) \in \{-1,1\}^{2n}} (\prod_{p=1}^{2n} x_p) \prod_{p=1}^{n} \sum_{i=1}^{2n-p-1} x_i x_{i+p+1} = 2^{2n+1} L(2,n)$$

The computation of $L(2,n)$ is in $O(4^n \times n^2)$ and an efficient long integer arithmetic is needed.

By using this approach, M. Godfrey has solved $L(2,20)$ in one week on three PCs in 2002. Later, Krajecki et al. [3] solved the $L(2,23)$ and the $L(2,24)$ problem instances, the latter in 3 months, using a dozen of computers. In spite of the evolution of CPUs processing power, the solution for the next problem instance - $L(2,27)$ - would require several months of intensive computing on a whole cluster.

II. Langford deployment on multiGPU clusters

Since the end of 2000's, GPUs become a fast and less expensive alternative to massive parallel computing on CPUs. The number of GPU cores that can be aligned in a single machine is much more expressive (for example, 2688 GPU cores in a Nvidia K20Xm Kepler GPU processor, against 16 cores in a Intel Xeon CPU). Nowadays supercalculators include GPUs, and multi-GPU architectures become more frequent in the TOP500 list.

One of the main limitations of GPUs is that their cores are simpler that CPU ones and the threads spread on these cores work synchronously. This prevents to take advantage of their potential on irregular applications, based on multiple tests and branchings.

Our approach uses a bottom-up tree (i.e., starting from the nth color) and combines Miller and Godfrey's techniques to efficiently perform the computation.

The Miller's tree search allows to prune the search space in order to highly reduce the search effort, but it is based on backtracking and thus cannot benefit of GPUs use. This step is therefore performed on CPU.

Our Miller's implementation is based on a binary representation of the "color" codes (for example, a color in level 1 has code "101", while a color in level 3 has code "10001") with bit-shifting and bit-wise XOR operations (Fig. 3).

Using a distributed computing middleware such as MPI or CloudFIT [4], the generated consistent masks

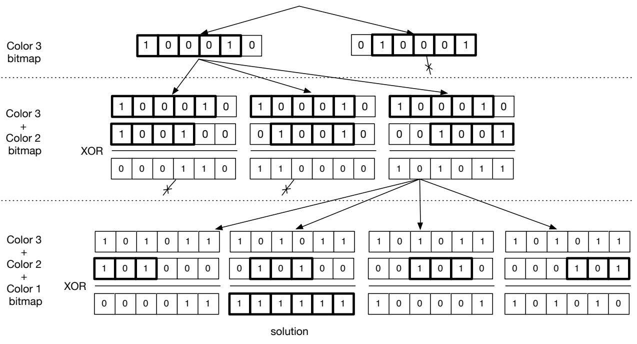

Figure 3. Bitmap evaluation of color alignments.

are treated by compute CPU-GPU clients as follows: the CPU prepares a large set of more refined masks in order to prepare grids and blocks to feed the GPU, which traverses the relative sub-tree with the regular Godfrey approach.

In order to take advantage of the GPUs compute efficiency, much effort has to be made to optimize the Godfrey implementation. In addition, code tuning imposes to fix the depth of the sub-trees to be considered by the GPU kernels; the depth dedicated to the clients grids generation is deduced from the server [Miller] masks generation and the GPU kernel [Godfrey] depth.

We have already developed the implementation of the Miller tasks generation, their distribution over the compute clients with client-server or cloud distribution, the generation of the sub-masks sets for GPU kernel computation, and a Miller regularized implementation of the kernel. The results prove our concept. The last effort to be done is to implement the GPU kernel version of the Godfrey approach, which we currently work on. We aim at the resolution of L(2,27) on the ROMEO cluster[2], a GPU-enhanced cluster ranked 151th on the TOP500 listing (Nov. 2013).

References

[1] M. Gardner, *Mathematics, Magic and Mystery*, 1956.

[2] J. E. Simpson, "Langford Sequences: perfect and hooked," *Discrete Math*, vol. 44, no. 1, pp. 97–104, 1983.

[3] C. Jaillet, M. Krajecki, and A. Bui, "Parallel tree search for combinatorial problems: a comparative study between openmp and mpi," *Studia Informatica Universalis*, vol. 4, no. 2, pp. 151–190, December 2005.

[4] L. Steffenel, O. Flauzac, A. S. Charao, P. P. Barcelos, B. Stein, S. Nesmachnow, M. K. Pinheiro, and D. Diaz, "Per-mare: Adaptive deployment of mapreduce over pervasive grids," in *Proceedings of the 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC'13)*, Compiegne, France, Oct 2013.

[2]https://romeo.univ-reims.fr/