Exactly solving permutation-based optimization problems on heterogeneous CPU/GPU clusters

Jan Gmys[1,2], Mohand Mezmaz[1], Nouredine Melab[2], Daniel Tuyttens[1]

[1] [1]University of Mons, Mathematics and Operations Research Department
{jan.gmys,mohand.mezmaz,daniel.tuyttens}@umons.ac.be
[2] [2]Inria Lille - Nord Europe, CNRS/CRIStAL, Université Lille 1, France
nouredine.melab@univ-lille1.fr

Introduction. The exact resolution of large-scale instances of combinatorial optimization problems (COPs) requires a huge amount of computational resources. The first exact resolution of *Ta056* [2], an instance permutation flow-shop scheduling problem (FSP) [3], illustrates the required computational effort. Using B&B@Grid, a B&B algorithm designed for computational grids, the optimal solution was found with proof of optimality within 25 days, exploiting on average 328 processors belonging to 9 distinct clusters of the nation-wide experimental testbed Grid'5000 [3]. In 2006, the most of the processors in Grid'5000 were either mono-core or dual-core CPUs. Today, according to the latest *Top500* (June 2017) ranking of the world's largest supercomputers [4], 93% of the Top500 systems use processors at least 8 cores and 27% use processors with 18 or more cores. On the road towards exascale computing, the ranking confirms the trend towards increasingly heterogeneous systems, as a total of 91 systems on the list use accelerator/co-processor devices, 80% of which are GPUs. In addition to their energy-efficiency, many-core devices have the potential to significantly boost the performance of traditional processors. This motivated us to revisit the design and implementation of B&B for hybrid multi-core and multi-GPU platforms, from single-node systems to large-scale heterogeneous high performance computing clusters. [1, 5, 6] Our study focuses on permutation-based COPs using the FSP, the Quadratic Assignment Problem (QAP) and the n-Queens puzzle problem as test-cases.

B&B for heterogeneous CPU/GPU clusters. The B&B algorithm perform an implicit enumeration of the search space by dynamically constructing and exploring a tree. This is done using four operators: branching, bounding, selection and pruning. The branching operator recursively decomposes the initial problem into smaller subproblems and a bounding function computes lower bounds on the optimal cost of these subproblems. The pruning operator uses these lower bounds to eliminate subproblems that cannot lead to an improvement of the best solution found so far. The selection operator guides the tree-traversal by returning the next subproblem to be processed according to a predefined strategy. B&B is a highly irregular application, both at the application level and at the instruction level. Because of the unpredictable pruning of branches, the shape of the explored tree is highly irregular. At a lower level, the combinatorial nature of evaluated subproblems leads to diverging control flows and irregular memory accesses. Most of the challenges one has to face when mapping B&B onto massively parallel architectures are immediate consequences of this irregularity. Namely, the main challenges we identified challenges include:

– the efficient storage and management of a huge number of subproblems, dynamically generated at runtime,
– the dynamic distribution of these subproblem among workers, i.e. load balancing a highly irregular application in a large-scale heterogeneous environment,
– the choice of the appropriate parallelization model, as the latter is strongly influenced by both the characteristics of the problem to be solved and the target execution platform,
– and the efficient exploitation of SIMD processing capabilities, a key performance lever of accelerator devices, at odds with he irregular nature of B&B.

Each of these challenges calls for answers that are, to a certain extent, hardware-specific and/or problem-specific. For the efficient storage and management we uses an innovative data structure

[3] https://www.grid5000.fr/
[4] http://www.top500.org

dedicated to permutation problems, called Integer-Vector-Matrix (IVM) [4] instead of conventional data structures (e. g. stacks, deques, priority queues). The principle of parallel IVM-based B&B is to have several independent B&B processes use their private IVM for the exploration of different parts of the search space. The latter are compactly encoded as intervals, using a one-to-one correspondence between the set of n-element permutations and the integer interval $[0, n![$. When the bounding operation is moderately time-consuming or when the latter is coprocessor-accelerated, the efficient management of subproblems is indeed a critical component of B&B. We demonstrate that IVM-based B&B algorithms can significantly outperform their counterparts based on conventional data structures. Also, the IVM data structure allows to implement the entire algorithm on GPUs, completely bypassing the CPU and thus reducing overhead from CPU-GPU data transfers.

The compact encoding of the search space can be used to design efficient load balancing mechanisms. In order to dynamically distribute the irregular B&B among processing cores, IVM-based B&B processes exchange intervals, used as work units, in a work stealing approach. However, the design of work stealing strategies must take hardware-related constraints (e. g. the availability of synchronization mechanisms) and the execution model (synchronous or asynchronous) into account. Thus, different, but compatible, work stealing mechanisms are proposed for multi-core-based and GPU-centric B&B. Using hierarchical work stealing approaches, inter-GPU and inter-node load balancing mechanisms for single-node multi-GPU systems and hybrid distributed clusters are presented.

Performance optimization at the instruction-level, i. e. efficient SIMD processing, is strongly problem-dependent. Targeting Intel multi-core and many-integrated core (MIC) processors, the CPU-based B&B uses a vectorized implementation of the FSP bounding operator. In order to improve control flow efficiency and reduce instruction replay overhead, the GPU-centric B&B uses mapping schemes specifically adapted to each B&B operator. Moreover, the GPU-centric B&B uses two different parallelization models, depending on the granularity of the problem being solved.

Experimental Evaluation. Experimental results demonstrate the scalability of the approach at different levels, with respect to the number of CPU cores, GPU cores, GPU devices and heterogeneous compute nodes. In particular the use of multi-GPU systems and large clusters allows to solve instances whose exact resolution is otherwise impractical. For a class of 20 jobs/20 machines FSP instances with sequential execution times between 15 minutes and 22 hours, the resolution time using four GPUs ranges from 1 second to 1 minute, i. e. an improvement of three orders of magnitude compared to a single CPU core. The aforementioned large FSP instance *Ta056*, defined by 50 jobs and 20 machines is solved within 9 hours on a cluster containing 36 Pascal P100 GPUs and 18 Power8+ CPU (for a total of 180 CPU cores and 130 000 GPU cores). Fig. 1 shows the execution time for instance *Ta056* on different computing platforms.

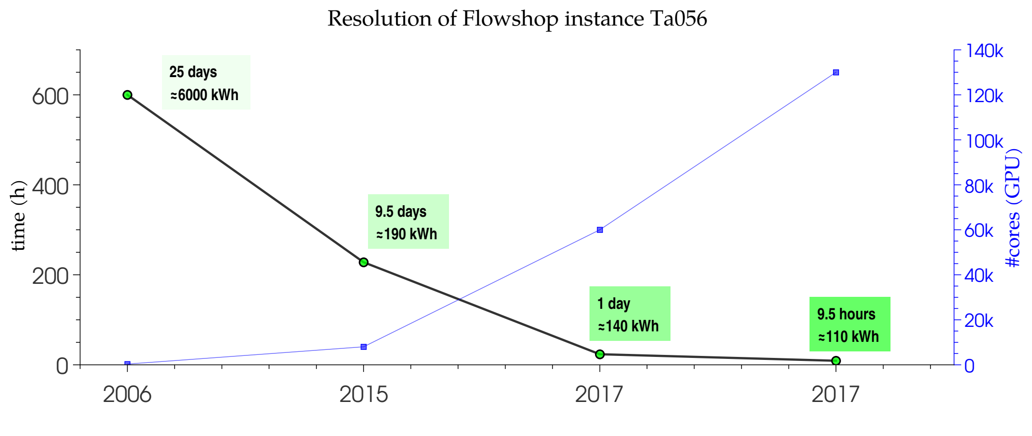

Fig. 1. Resolution of FSP instance *Ta056* (6.5×10^{12} evaluated nodes). **2006**: B&B@Grid ([2]), **2015**: multi-GPU-B&B(4xGTX980), **06/2017**: distributed GPU-B&B (16xGTX1080Ti), **08/2017** hybrid distributed B&B (36xP100 + 18xPower8+).

References

1. Gmys J., Mezmaz M., Melab N., Tuyttens D.: IVM-Based parallel branch-and-bound using hierarchical work stealing on multi-GPU systems in Concurrency & Computation : Practice & Experience (2016), 29(9)
2. Mezmaz M., Melab N., Talbi E. G.: A grid-enabled branch and bound algorithm for solving challenging combinatorial optimization problems, in 2007 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Mar. 2007, pp. 1–9
3. Taillard, E.: Benchmarks for basic scheduling problems", Journal of Opera- tional Research, vol. 64, pp. 278–285, (1993).
4. Mezmaz M., Leroy R., Melab N., and Tuyttens D.: A Multi-Core Parallel Branch-and-Bound Algorithm Using Factorial Number System, in 28^{th} IEEE International Parallel and Distributed Processing Symposium (IPDPS), May 2014, pp. 1203–1212
5. Gmys J., Leroy R., Mezmaz M., Melab N., Tuyttens D.: Work stealing with private integer-vector-matrix data structure for multi-core branch-and-bound algorithms, in Concurrency & Computation : Practice & Experience,28, 18, 4463-4484 (2016)
6. Melab N., Gmys J., Mezmaz M., Tuyttens D.: Multi-core versus many-core computing for many-task Branch-and-Bound applied to big optimization problems, in Future Generation Computer Systems (2017), in press