CONCURRENTLY EMPLOYING RESOURCES OF SEVERAL SUPERCOMPUTERS WITH PARASCIP SOLVER BY EVEREST PLATFORM

S.A. Smirnov[1,a], V.V. Voloshinov[1], O.V. Sukhoroslov[1,2]

[1] *Institute for Information Transmission Problems of the Russian Academy of Sciences (Kharkevich Institute), Bolshoy Karetny per. 19, build.1, Moscow, 127051, Russia*

[2] *HSE University, Moscow, Russia*

E-mail: [a] sasmir@gmail.com

ParaSCIP is rather advanced open-source solver for discrete and global optimization problems. This solver is distinguished by that it can run on distributed memory systems and use up to 80,000 cores, solving open problems from the MIPLIB test libraries. Earlier, using this solver, we confirmed the conjecture on optimal packing of nine congruent circles on a square flat torus. The goal of the study was to increase computing performance by utilizing resources of multiple clusters to solve hard optimization problem. To do this, we use the previously developed DDBNB application, which allows to speed up the solution of optimization problems by using coarse-grained parallelization based on a static decomposition of feasible domain made before solving starts. DDBNB is an application for the Everest distributed computing platform which is responsible for running jobs on heterogeneous computing resources (servers, cloud instances, clusters, etc.). As a result, DDBNB, Everest, and ParaSCIP had to be modified to make it possible to exchange incumbents (feasible solutions found by the solver) between several ParaSCIP instances running on different supercomputers. The resulting system was benchmarked using three different instances of Traveling Salesman Problem. The supercomputers HPC5 of the NRC "Kurchatov Institute" and cHARISMa of the HSE University were used as computing resources. As a result, for two problem instances, there is an effect, and the speedup is especially noticeable for a more complex problem. However, for a simpler problem, the exchange of incumbents does not seem to affect the amount of speedup. For the third instance, there is no particular effect, at least no slowdown is observed.

Keywords: distributed computing, DDBNB, ParaSCIP, branch-and-bound, domain decomposition, global optimization, discrete optimization, message passing interface

Sergey Smirnov, Vladimir Voloshinov, Oleg Sukhoroslov

1. Introduction

DDBNB [1, 2, 3] is a distributed application processing a set of computational tasks in parallel $\{T_k : k = 1 : K\}$. Each of *T* is a pair {*P*, *S*}, where *P* is the initial data of some mathematical programming problem (represented as NL-file), *S* is a set of solver settings. Each *P* is a subproblem of some main problem in the sense that its set of feasible solutions is a subset (possibly empty) of the feasible domain of the main problem. The key features of DDBNB are: 1) immutability of the specified set of tasks; 2) the minimum data exchange between these tasks (only incumbent solution values). This is consistent with the coarse-grained parallelism model. DDBNB is an application for the Everest distributed computing platform [4], http://everest.distcomp.org, which is responsible for running jobs on heterogeneous computing resources (servers, clusters, clouds, etc.).

SCIP [5] is a state-of-the-art branch-and-bound solver for various types of discrete and global optimization problems. In the context of this work, it is important that SCIP allows solving mixed-integer linear and nonlinear mathematical programming problems (MILP, MINLP). It is currently the fastest [6] and most advanced (in terms of the ability to solve different types of problems, the frequency of new versions and the composition of the development team) open source solver.

SCIP is distributed as a part of SCIP Optimization Suite, which includes the UG framework [7] to enable SCIP parallelization. There are two parallel solvers based on UG: FiberSCIP, a multi-threaded application (for shared memory environment), and parallel MPI application ParaSCIP, which could be launched on a computational cluster. In ParaSCIP MPI processes are not equivalent: there is one dedicated Load Coordinator (LC) and multiple solver processes. On the Load Coordinator's side, a presolve is made and then presolved problem is distributed to solver processes. Then the so-called Ramp Up occurs, when the solver processes begin to solve the problem with the aim of spawning new subproblems and transferring them to the LC so that it has enough subproblems to load all the other work processes. Solver processes and the LC exchange mainly small messages with the incumbent solution values. Thus, the key features of ParaSCIP are: 1) subtasks are generated dynamically, without requiring any manual action from the user; 2) the amount of data exchanged between MPI processes is similar to DDBNB's one; 3) ParaSCIP works only on clusters.

In this study, an attempt was made to use more computational resources at the same time with ParaSCIP by utilizing resources of multiple clusters in order to solve problem instances faster. To do this, we use the DDBNB application we previously developed. As a result, DDBNB, Everest, and ParaSCIP had to be modified to make it possible to exchange incumbent solutions (feasible solutions found by the solver) between several ParaSCIP instances running on different supercomputers.

2. UG/ParaSCIP, DDBNB and Everest Modification

The process of integrating ParaSCIP solver into DDBNB is different from the integration of the previously added CBC and SCIP [1, 2, 3] solvers. ParaSCIP is a parallel application that runs on a computing cluster and uses MPI (Message Passing Interface) to exchange data between solver processes and a Load Coordinator process [fig. 1]. Thus, launching this solver differs significantly from launching ordinary solvers like SCIP and CBC, which operate within a single process. DDBNB was modified to allow launching ParaSCIP as MPI application.

To load an incumbent solution into ParaSCIP, not only the objective function value of the solution are required, but also the values of the decision variables for this solution [8]. Earlier, data exchange between subtasks in DDBNB implied transmission of objective function values only. These values could be obtained and changed through messages sent to the Everest server through the so-called task messages — special messages by which a task launched by an Everest agent can interact with the Everest server [fig. 2]. For the exchange of incumbent solutions with values of variables, the task message format in Everest has been modified: an additional parameter was added to task messages so that it became possible to transmit a text string with encoded values of non-zero decision variables along with the incumbent value.

Similarly to other solvers, the most complicated part was to modify ParaSCIP so that it could receive and use an incumbent solution found by another instance of the solver (for example, running on another cluster). When solver finds a new incumbent solution, our code converts it to original problem coordinates, saves it to a file on disk and reports file's name to a separate process which does data exchange with Everest server. When a new incumbent solution is received from outside, our code reads it, converts and sends it to ParaSCIP's main process (Load Coordinator [fig. 1]) with MPI_Isend.

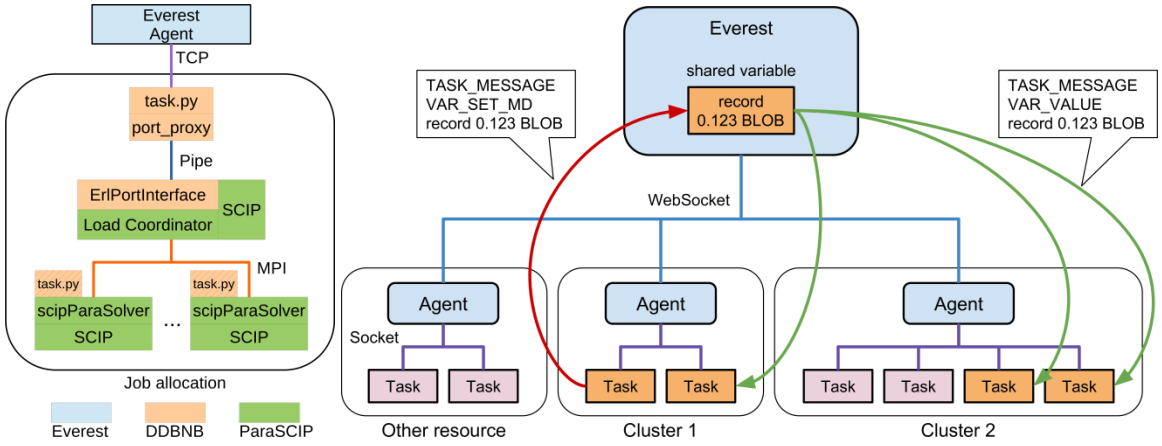

Figure 1 (left). Application processes and types of communication between them within a computing cluster. Figure 2 (right). Exchanging complete solutions within Everest.

3. Computational Experiments

The goal of the computational experiments was to check the possibility of exchanging incumbents over the network for tasks running on several computational clusters simultaneously and to assess its effect on the solving time of optimization problems.

For the experiments, the Traveling Salesman Problem was chosen with three data sets from TSPLIB [9]: ch150, ch130 and bier127. From each data set, two sets of AMPL NL-files were created: single NL (dd0) corresponding to the original problem, and four NLs (dd2) corresponding to four subproblems obtained by fixing two boolean variables. AMPL model was described in [10].

The supercomputers HPC5 of the NRC "Kurchatov Institute" and cHARISMa of the HSE University were used as computing resources. Most of the tests were carried out on the cluster of the HSE University, as it was less loaded, which made it possible to obtain more stable figures for performance tests. Two clusters were used only in the integration experiment to show the possibility of simultaneous work and exchange of incumbents on two different clusters.

For the computational experiment, the ParaSCIP solver from the SCIP 6.0.2 distribution with the aforementioned modifications was built on both clusters. To run DDBNB with ParaSCIP, a separate Everest agent was configured on the clusters using the basic "local" computing resource adapter, which allows to fork processes on the cluster's submission node. We did not use specialized adapter with SLURM support [2] which allows running on a cluster only batches of jobs with one process each, because with ParaSCIP it is necessary to request the cluster's resource manager to allocate resources with several processes within one job.

Table 1 lists the series of DDBNB executions made with ParaSCIP. In each series, 10 DDBNB executions were made and the results were taken only for those where all tasks started running on clusters approximately simultaneously (the difference between the start times of the last and the first is no more than 60 seconds). The number of runs that meet this condition is listed in the "N runs" column. The median run time (the end time of the last task minus the start time of the first) is shown in the "Median" column. These medians are shown in [fig. 3].

Table 1. DDBNB with ParaSCIP runs

Model	Label	Comm.	N fixed	N tasks	N proc	Task location	N runs	Median	Stdev
ch150	dd0	-	0	1	32	1xHSE	10	5414	3792
ch150	dd2	+	2	4	32	4xHSE	8	2591	868
ch150	dd2-n/c	-	2	4	32	4xHSE	5	8692	2290
ch150	dd2+kiae	+	2	4	32	2xHSE 2xKIAE	9	5223	2672
ch150	dd0-kiae	-	0	1	32	1xKIAE	10	6731	2873
ch130	dd0	-	0	1	16	1xHSE	9	1604	470
ch130	dd2	+	2	4	16	4xHSE	10	1152	386
ch130	dd2-n/c	-	2	4	16	4xHSE	10	1143	284
bier127	dd0	-	0	1	16	1xHSE	10	820	163
bier127	dd2	+	2	4	16	4xHSE	10	840	48
bier127	dd2-n/c	-	2	4	16	4xHSE	10	1090	773

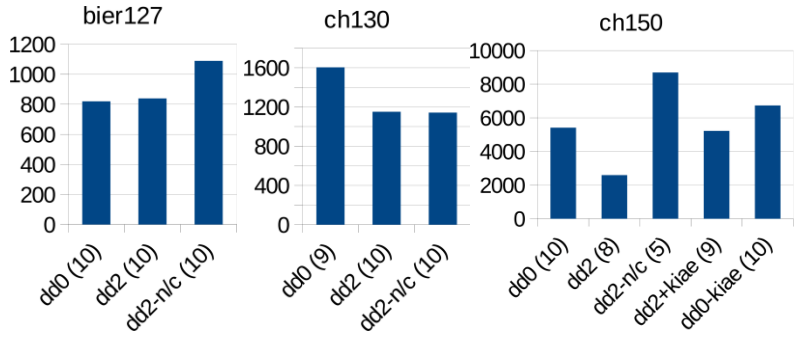

Figure 3. Median solution times in seconds for bier127, ch130 and ch150 datasets

To test whether the use of DDBNB with ParaSCIP gives a speedup, we compared the median runtimes for the dd0 and dd2 series for each of the three problems. As a result, there is speedup for ch150 and ch130 data sets. There is no slowdown for bier127.

To check if the speedup can be caused by simple dividing the problem into subproblems, we compare the dd0 and dd2-n/c (with the incumbents exchange disabled) series. So, the problem was solved without decomposition and with decomposition into four subproblems, respectively, but without exchanging incumbents. It turns out that for ch150 and bier127 this scenario gives a slowdown, and for ch130 it gives performance comparable to the solution with the incumbents exchange (dd2).

Comparison of the dd2 and dd2-n/c series allows us to evaluate the effect of the exchange of incumbents. It can be seen that there is no effect for ch130, and for ch150 and bier127, the exchange of incumbents gives a speedup, although for the latter it does not allow to become faster than solving the original problem with a single instance of ParaSCIP.

For the ch150 task [fig. 3], launches were made on two clusters at once: it can be seen that the dd2+kiae series, where two subtasks were executed on one cluster and two on the other, worked faster than the dd0-kiae series and about the same as the dd0 series, but noticeably faster than dd2-n/c. That is, the exchange of incumbents for tasks simultaneously running on two different clusters made it possible to speed up, although not as much as the execution of all four subtasks on the faster and less loaded cluster gives.

4. Conclusion and Future Work

The paper presents the modification to the DDBNB application, the Everest cloud platform and the ParaSCIP solver which were improved to allow the exchange of incumbents between several instances of ParaSCIP running in different jobs on clusters. DDBNB with modifications and a patch

for UG framework are available in a separate branch in a DDBNB's GitHub repository, https://github.com/distcomp/ddbnb/tree/parascip. Computational experiments have been carried out showing the effect of the improvements made. There is no particular effect for one of the three problem instances, at least no slowdown is observed. For the other two, there is an effect, and the speedup is especially noticeable for a more complex problem (ch150). However, for a simpler problem (ch130), the exchange of records does not seem to affect the amount of speedup. Thus, we can conclude that the integration of DDBNB and the ParaSCIP solver was successful, but its effect is not stable. Future work could be focused on improving usability of the presented approach by allowing setting solver parameters and resource manager settings.

5. Acknowledgement

Authors thank Stefan Vigerske and Yuji Shinano for their consultations on using of SCIP and ParaSCIP solvers. This work is supported by the Russian Science Foundation (Project 20-07-00701). This research was supported in part through resources of supercomputer facilities provided by NRU HSE. This work has been carried out using computing resources of the federal collective usage centre Complex for Simulation and Data Processing for Mega-science Facilities at NRC "Kurchatov Institute" (ministry subvention under agreement RFMEFI62117X0016), ckp.nrcki.ru.

References

[1] S. Smirnov and V. Voloshinov. On domain decomposition strategies to parallelize branch-and-bound method for global optimization in Everest distributed environment. Procedia Computer Science, 136:128–135, 2018.

[2] S. Smirnov, O. Sukhoroslov, and V. Voloshinov. Using resources of supercomputing centers with everest platform. In V. Voevodin and S. Sobolev, editors, Supercomputing, Communications in Computer and Information Science, pages 687–698. Springer International Publishing, 2018.

[3] S. Smirnov, V. Voloshinov. Implementation of concurrent parallelization of branch-and-bound algorithm in Everest distributed environment. Procedia Computer Science, 119:83–89, 2017.

[4] O. Sukhoroslov, S. Volkov, A. Afanasiev. A web-based platform for publication and distributed execution of computing applications. 2015 14th International Symposium on Parallel and Distributed Computing, pp. 175-184, 2015.

[5] A. Gleixner, M. Bastubbe, L. Eifler, T. Gally, G. Gamrath, R. L. Gottwald, G. Hendel, C. Hojny, T. Koch, M. E. Lübbecke, S. J. Maher, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, F. Schlösser, C. Schubert, F. Serrano, Y. Shinano, J. M. Viernickel, M. Walter, F. Wegscheider, J. T. Witt, J. Witzig. The SCIP Optimization Suite 6.0. ZIB-Report 18-26, Zuse Institute Berlin, July 2018.

[6] H. Mittelmann. The MIPLIB2017 Benchmark Instances. Available at: http://plato.asu.edu/ftp/milp.html (accessed 14.09.2021)

[7] Y. Shinano, T. Achterberg, T. Berthold, S. Heinz, and T. Koch. Parascip – a parallel extension of scip. In Christian Bischof, Heinz-Gerd Hegering, Wolfgang E. Nagel, and Gabriel Wittum, editors, Competence in High Performance Computing 2010, pages 135–148. Springer, 2012.

[8] Sergey Smirnov and Vladimir Voloshinov. Integration of ParaSCIP solvers running on several clusters on the base of everest cloud platform. Procedia Computer Science, 156:13–18, 2019.

[9] MP-TESTDATA - The TSPLIB Symmetric Traveling Salesman Problem Instances. Available at: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/ (accessed 14.09.2021)

[10] V. Voloshinov, S. Smirnov, O. Sukhoroslov. Implementation and use of coarse-grained parallel branch-and-bound in Everest distributed environment. Procedia Computer Science, 108:1532–1541, 2017.