Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 156 (2019) 13–18

Procedia Computer Science

www.elsevier.com/locate/procedia

8th International Young Scientist Conference on Computational Science

Integration of ParaSCIP solvers running on several clusters on the base of Everest cloud platform

Sergey Smirnov[a,*], Vladimir Voloshinov[a]

[a]*Institute for Information Transmission Problems of the Russian Academy of Sciences, Bolshoy Karetny per. 19-1, Moscow 127051, Russia*

Abstract

Software integration of optimization problems' solvers leveraging power of heterogeneous computing environments is a great challenge of last decades. Last several years we have been developing coarse-grained parallelization approaches to speed up Branch-and-Bound (BnB) algorithm for discrete and global optimization problems by exchange of BnB-incumbents in a heterogeneous environment containing standalone servers and clusters via Everest software toolkit, http://everest.distcomp.org. This approach have been implemented as DDBNB Everest-application (Domain Decomposition BnB), https://github.com/distcomp/ddbnb. The current implementation is based on two solvers (and their open API to get/put incumbents): SCIP, https://scip.zib.de and CBC, https://github.com/coin-or/Cbc. Recently we began to use ParaSCIP solver, https://ug.zib.de, - parallel implementation of BnB-algorithm in SCIP based on MPI. ParaSCIP demonstrates an advantages of fine-grained parallelization in homogeneous computing environment, i.e. HPC-clusters. By now we have access to three clusters from Russian Top50 where ParaSCIP have been installed. In the article several ways to involve ParaSCIP processes running on different clusters in solving common optimization problem are discussed.

Keywords: branch-and-bound; domain decomposition; distributed global optimization; message passing interface

1. Introduction

ParaSCIP is a parallel extension of state-of-the-art optimization solver SCIP on the base of UG framework, which uses MPI technology. ParaSCIP has solved some of the open instances in MIPLIB 2010 library for the first time. DDBNB is an implementation of parallel branch-and-bound algorithm using coarse-grained parallelism. We have successfully used ParaSCIP in experiments with a number of combinatorial geometry problems (Tammes and Thomson problems, Packings of Congruent Circles on a Square Flat Torus) [1, 2].

We currently work with three supercomputers located in different institutions (see Table 1 for details):

* Corresponding author. Tel.: +0-000-000-0000 ; fax: +0-000-000-0000.
E-mail address: sergey.smirnov@iitp.ru

1877-0509
10.1016/j.procs.2019.08.124

1. “Lomonosov”, MSU, https://users.parallel.ru/wiki/pages/22-config;
2. HPC4, NRC ”Kurchatov Institute”, http://computing.nrcki.ru;
3. “Govorun”, LIT JINR, http://hlit.jinr.ru/about_govorun/cpu_govorun.

This clusters are shared between many users and are usually occupied to some degree. For calculations utilising ParaSCIP it’s good to have many processes available to traverse search tree effectively. And it’s hard to gather enough resources on just one cluster: the job gets queued for a long time. Also long running jobs utilising many nodes are discouraged because this makes the supercomputer unavailable to some degree to other users. On the other side using resources of multiple clusters can provide much more computing power than a single cluster.

There are multiple approaches to utilising multiple supercomputers with ParaSCIP. Let us briefly describe them:

1. Use some MPI implementation which allows to run an MPI application transparently on resources provided by different job allocations of multiple clusters.
2. Execute ParaSCIP on a single supercomputer to get a checkpoint file with multiple nodes (subtrees) saved. Then split this file in parts and execute independent ParaSCIP instances with new checkpoint files one per cluster.
3. Modify ParaSCIP so that worker processes could be added and removed dynamically and could be accessed using a proxy.
4. Use DDBNB with Everest platform [3] to do some initial decomposition of the problem manually and then solve subproblems on multiple ParaSCIP instances running on separate supercomputers. Modify ParaSCIP to implement incumbent exchange through DDBNB.

Table 1. Characteristics of partitions used on “Lomonosov”, HPC4 and “Govorun”

Cluster	Partition	CPU	cores per node	mem. per node
“Lomonosov”	regular4	Intel®Xeon®X5570 2.93GHz	8	12 Gb RAM
HPC4	hpc4-3d	Intel®Xeon®E5-2680 v3 2.50GHz	24	128 Gb RAM
“Govorun”	skylake	Intel®Xeon®Gold 6154 3.00–3.70GHz	36	128 Gb RAM

2. Possible ways to increase performance of BnB solvers in distributed computing environment

There are two basic (from the mathematical point of view) approaches to leverage performance of BnB-solvers in distributed computing environment (see e.g. [4, 5, 6]): decomposition of a feasible domain (DD) and concurrent optimization (CO). Decomposition of feasible domain into subsets gives a set of appropriate subproblems which may be sent to a pool of solvers running in parallel. Concurrent optimization means solving the same problem by a pool of BnB-solvers running in parallel with different settings of BnB-algorithm. Up-to-date BnB-solvers has hundreds of options dozens of which define rules of BnB-search-tree traversing (e.g. depth-first search, breadth-first search), node selection rule for branching (by pseudo-cost, by low bound, some other heuristic), etc. For both approaches (CO and/or DD), performance may be increased via exchange of so called “incumbents” (best values of goal function on feasible solutions) found by solvers’ processes running in parallel. Besides “mathematical” aspect, from programmatic point of view, there are two main ways of implementation of BNB-algorithm in distributed environment appear definable: fine-grained parallelization (FG) and coarse-grained one (CG).

Important features of FG-approach are dynamical generation of subproblems (for DD) or another instances of already existing (sub)problems (for CO) and their distribution among pool of “worker”-solvers. Usually, one dedicated Load Coordinator process used to monitor states of subproblems solving and redistribute new subproblems among running solvers (e.g. see [7]). Another peculiarities are rather intensive data flow among running processes involved in solving and, hence, high requirements to computing environment (homogeneous is preferable) and to quality of middle-ware to be used in programmatic implementation (MPI is common choice).

On the other hand, CG-approach usually based on preliminary (“static”) generation of subproblems in accordance with some heuristic rules: by domain decomposition and/or defining sets of options for concurrent optimization