Future Generation Computer Systems 42 (2015) 44–54

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Elastic grid resource provisioning with WoBinGO: A parallel framework for genetic algorithm based optimization

Milos Ivanovic [a,*], Visnja Simic [a], Boban Stojanovic [a], Ana Kaplarevic-Malisic [a], Branko Marovic [b]

[a] Faculty of Science, University of Kragujevac, Radoja Domanovica 12, 34000 Kragujevac, Serbia

[b] School of Electrical Engineering, University of Belgrade, Bulevar kralja Aleksandra 73, 11120 Belgrade, Serbia

HIGHLIGHTS

- Framework for optimization using parallel GA over Grid and HPC resources.
- Provides elastic resource provisioning avoiding unnecessary occupation of resources.
- Automatic adaptive allocation of jobs with limited lifetime.
- Limited job lifetime provides friendliness towards other batching queue users.
- The complexity of underlying Grid infrastructure is hidden from the user.

ARTICLE INFO

Article history:
Received 9 December 2013
Received in revised form
20 March 2014
Accepted 2 September 2014
Available online 16 September 2014

Keywords:
Grid computing
Pilot-job infrastructure
Dynamic resource provisioning
Metaheuristics based optimization framework

ABSTRACT

In this paper, we present the WoBinGO (**Wo**rk **B**inder **G**enetic algorithm based **O**ptimization) framework for solving optimization problems over a Grid. It overcomes the shortcomings of earlier static pilot-job frameworks, by: (1) providing elastic resource provisioning thus avoiding unnecessary occupation of Grid resources; (2) providing friendliness towards other batching queue users thanks to adaptive allocation of jobs with limited lifetime. It hides the complexity of the underlying Grid environment, allowing the users to concentrate on the optimization problems. Theoretical analysis of possible speed-up is presented. An empirical study using an artificial problem, as well as a real-world calibration problem of a leakage model at the Visegrad power plant were performed. The obtained results show that despite WoBinGO's adaptive and frugal allocation of computing resources, it provides significant speed-up when dealing with problems that have computationally expensive evaluations. Moreover, the benchmarks were performed in order to estimate the influence of the limited job lifetime feature on the queuing time of other batching jobs, compared to a static pilot-job infrastructure.

1. Introduction

Grid computing [1] has emerged as an effective environment for the execution of parallel applications that require great computing power. Grid computing consists of a geographically distributed infrastructure gathering computer resources around the world in a transparent way. Users are provided access to enormous computing resources, and enabled to better meet the challenges of science and engineering. One of the most frequently encountered challenges in applied science and engineering, is optimization. Genetic algorithms (GAs) [2] have proven themselves as robust and powerful mechanisms when it comes to solving complex real-world optimization problems. GA is characterized by a large number of function evaluations. Due to the time-consuming fitness evaluation functions found in real-world problems, it may take days and months for the GA to find an acceptable solution. Speeding up the optimization process is achieved by parallelization of GA [3–5] which reduces the resolution times to reasonable levels. Grid computing environments provide the infrastructure for implementing parallel metaheuristics. There, researchers face new difficulties associated with developing and deploying a Grid based application. The fact that Grid resources are distributed, heterogeneous and non-dedicated, makes writing parallel Grid-aware applications very challenging [6]. The development and

* Corresponding author. Tel.: +381 34 336223; fax: +381 34 335040.
E-mail addresses: mivanovic@kg.ac.rs (M. Ivanovic), visnja@kg.ac.rs (V. Simic), bobi@kg.ac.rs (B. Stojanovic), ana@kg.ac.rs (A. Kaplarevic-Malisic), branko.marovic@rcub.bg.ac.rs (B. Marovic).

http://dx.doi.org/10.1016/j.future.2014.09.004
0167-739X/

execution of Grid applications require considerable effort and expert knowledge. Understanding the basics of Grid computing and Grid middlewares is a time and energy consuming process for developers. Moreover, for each run of application on the Grid, one has to address the issues of Grid resource discovery and selection, Grid job preparation, submission, monitoring and termination which differ from one middleware to another. These differences among middlewares may limit or hinder portability between different Grid infrastructures. Aside from the complexity of the Grid infrastructures, certain limitations are also present, notably the need for users to wait, sometimes for a significant time until their requests for computing resources are processed and the lack of good support for interactive applications. The complexity involved in writing Grid-enabled applications averts researchers from harnessing computational Grids by scientific applications. As Grids grow in size at an admirable rate and an increasing number of resources are put at Grid users' disposal, it is of utmost importance for the researchers to efficiently exploit computational Grids in order to solve real-world problems. In this context, tools for simplifying Grid application development, by hiding the complexity of Grid computing from the researchers, can significantly enhance Grids harnessing by scientific and engineering applications.

In this paper, we present the WoBinGO framework for solving optimization problems over heterogeneous resources, including HPC clusters and Globus-based Grids. Although it is possible to utilize diverse computing resources for solving optimization problems in parallel (multiple university clusters, Grid), having in mind the immense computing power offered by the Grid, we will restrict our discussion in this paper only to EGI (European Grid Initiative) deployment of WoBinGO. The framework was designed to meet the following goals: (1) speeding up the optimization process by parallelization of GA over the Grid; (2) relieving the researcher burden of obtaining Grid resources and dealing with various Grid middlewares; (3) enabling fast allocation of Grid jobs to avoid waiting until requests for computing resources are processed by Grid middleware; (4) providing flexible allocation of worker jobs in accordance with the dynamics of the users' requests, thus avoiding the unnecessary reservation of computing resources.

The framework is dedicated for parallel execution of single and multi-objective optimization using GA on the Grid. It uses a master–slave parallelization model and allows both: parallel evaluation of a population in GA and parallel execution of multiple instances of the parallel GA. As a novelty, this framework incorporates the Work Binder (WB) [7] which provides almost instant access to Grid resources and interactivity for client applications. Integration of WB into the framework enables the programmer to focus solely on the optimization problem without having to worry about specific details of Grid computing. Additionally, WB increases the utilization of the Grid infrastructure by offering automated elasticity in its occupancy, based on present and recent client behaviour. Furthermore, a single WB service is capable of serving multiple users with multiple GA instances, where for each instance of GA a population evaluation is also parallelized. Due to the multi-tier design, it is easily possible to replace master–slave parallelization model with hierarchical parallel GA with master–slave demes [8] or with PEGA (parallel cellular GA) [9], keeping all the other components intact. The framework also adheres to the standard Globus security mechanisms, including GSI and MyProxy.

With the master–slave parallelization model and WB, evaluation of individuals is separated from the rest of the algorithm and performed on Grid computing elements (CEs). This allows an objective function to be written in any compiled or script language, which makes our framework favourable for solving optimization problems in diverse areas of science and engineering. The framework has been developed as an effort to efficiently solve optimization problems from the field of hydrology, but can be used for any other optimization task suitable for GA treatment.

Benchmarks were carried out using EMI/UMD middleware [10] on the South-East European regional infrastructure in order to evaluate the usability and efficiency of the proposed framework. The obtained results show that the achieved speed-up is almost linear. Moreover, the benchmarks were performed in order to estimate the influence of the limited job lifetime feature on the queuing time of other batching jobs. Compared to a static pilot-job infrastructure, this waiting time was significantly reduced. Further details about pan-European Grid, aspects such as production infrastructure, the management tools and the operational services offered can be found in [11]. The process of building regional Grid infrastructure is thoroughly described in [12].

The rest of the paper is organized as follows: in Section 2, we review the related work. A description of the framework is given in Section 3. In Section 4, theoretical analysis of WoBinGO's speed-up along with the experimental results and discussion are given. A case study is presented in Section 5, followed by concluding remarks in the last section.

2. Related work

Grid oriented genetic algorithms (GOGAs, following the notation first introduced by [13]) have been used over the past years for solving different problems [14–16]. The research community has also proposed and implemented optimization frameworks with parallel metaheuristics included. Most of them have been using small, dedicated and homogeneous computing resources. Here, we will only discuss those that enable execution of parallel metaheuristics in Grid computing environments.

GridUFO is a service oriented optimization framework [17] that offers sharing of optimization algorithms and problems among GridUFO users and solving of optimization problems using an algorithm already registered with the framework. New algorithms and objective functions can be registered with the framework, but only C language code is acceptable. This is a huge limitation since the hydrocodes of our main interest are written in C#/.NET and Fortran. The authors report significant speed-up for the problems of a larger size, but they do not consider the time spent for scheduling the Grid job.

ParadisEO-CMW [18] is the framework for designing and deploying parallel metaheuristics on computational Grids, assembling together the ParadisEO [19] and MW [20] frameworks. Grid-enabling an application with MW involves the reimplementation of a number of virtual functions. The framework is only intended for Grids consisting of multiple Condor pools combined via flocking.

JG^{2}A [21] was created as an extension of JGA [22] to take advantage of Grid technologies, allowing population evaluation parallelization and parallelization of the GA parameter tuning experiments. JG^{2}A uses GT4 Grid middleware, but requires Condor as an underlying scheduler. This makes it inflexible because, in general, local resource managers other than Condor are used at different computing sites and these sites may be under a different administrative control, which makes it hard to enforce the deployment of Condor on all these sites.

Efficient hierarchical parallel GA framework using Grid computing (GE-HPGA) [23] hides the complexity of a Grid environment through the extended GridRPC API and a metascheduler for automatic resource discovery. It has a two level structure: at the first level, sub-populations are transferred onto remote computing clusters and subpopulation evolution is invocated using the Globus