Future Generation Computer Systems 29 (2013) 1682–1699

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

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

A family of heuristics for agent-based elastic Cloud bag-of-tasks concurrent scheduling

J. Octavio Gutierrez-Garcia, Kwang Mong Sim *

[a] Department of Information and Communications, Gwangju Institute of Science and Technology, Gwangju 500-712, Republic of Korea
[b] School of Computing, University of Kent, Medway Building, Chatham Maritime, Kent ME4 4AG, United Kingdom

ARTICLE INFO

Article history:
Received 15 June 2011
Received in revised form
2 November 2011
Accepted 18 January 2012
Available online 7 February 2012

Keywords:
Scheduling heuristics
Bag-of-tasks applications
Cloud computing
Multi-agent systems
Agent-based Cloud computing
Elastic Cloud resource allocation

ABSTRACT

The scheduling and execution of bag-of-tasks applications (BoTs) in Clouds is performed on sets of virtualized Cloud resources that start being exhausted right after their allocation disregarding whether tasks are being executed. In addition, BoTs may be executed in potentially heterogeneous sets of Cloud resources, which may be either previously allocated for a different and fixed number of hours or dynamically reallocated as needed. In this paper, a family of 14 scheduling heuristics for concurrently executing BoTs in Cloud environments is proposed. The Cloud scheduling heuristics are adapted to the resource allocation settings (e.g., 1-hour time slots) of Clouds by focusing on maximizing Cloud resource utilization based on the remaining allocation times of Cloud resources. Cloud scheduling heuristics supported by information about BoT tasks (e.g., task size) and/or Cloud resource performances are proposed. Additionally, scheduling heuristics that require no information of either Cloud resources or tasks are also proposed. The Cloud scheduling heuristics support the dynamic inclusion of new Cloud resources while scheduling and executing a given BoT without rescheduling. Furthermore, an elastic Cloud resource allocation mechanism that autonomously and dynamically reallocates Cloud resources on demand to BoT executions is proposed. Moreover, an agent-based Cloud BoT scheduling approach that supports concurrent and parallel scheduling and execution of BoTs, and concurrent and parallel dynamic selection and composition of Cloud resources (by making use of the well-known contract net protocol) from multiple and distributed Cloud providers is designed and implemented. Empirical results show that BoTs can be (i) efficiently executed by attaining similar (in some cases shorter) makespans to commonly used benchmark heuristics (e.g., Max–min), (ii) effectively executed by achieving a 100% success execution rate even with high BoT execution request rates and executing BoTs in a concurrent and parallel manner, and that (iii) BoTs are economically executed by elastically reallocating Cloud resources on demand.

1. Introduction

Bag-of-tasks applications (BoTs) are sets of numerous unconnected (i.e., without precedence constraints) tasks, which can be highly parallelized given their unconnected nature. Scheduling and executing BoTs in Cloud environments is performed on sets of virtualized Cloud resources, which are allocated in terms of fixed and predefined allocation slots (e.g., 1-hour time slots in Amazon EC2 [1]) that start being exhausted right after their allocation disregarding whether tasks are being executed. In addition, BoTs may be executed in potentially heterogeneous sets of Cloud resources, which may be either previously allocated for a different and fixed number of hours or dynamically reallocated as needed (taking advantage of the elasticity of Cloud environments).

The scheduling of independent tasks in a set of heterogeneous computing resources has been shown to be an NP-complete problem [2]. For this reason, many heuristics have been proposed, from low level execution of tasks in multiple processors [3,4] to high level execution of tasks in Grid and Cloud environments [5–9]. Scheduling heuristics can be classified into: immediate and batch mode scheduling heuristics [10]. Immediate mode scheduling heuristics map BoT tasks to Cloud resources as soon as they arrive at the scheduler, e.g., the first-come–first-served scheduling heuristic. Batch mode scheduling heuristics pre-schedule BoT tasks on a previously defined set of Cloud resources before starting the execution, e.g., the Min–min and Max–min scheduling heuristics (first proposed in [4] and implemented in Grid-like settings in [11]). However, the majority of the scheduling heuristics (both immediate and batch modes) presumes that BoT holders are only charged for task execution time or not charged at all, when in

* Corresponding author.
E-mail addresses: joseogg@gmail.com (J.O. Gutierrez-Garcia), prof_sim_2002@yahoo.com (K.M. Sim).

0167-739X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2012.01.005

Cloud environments, consumers are charged for complete (1-hour) allocation slots.

In this paper, a family of 14 Cloud scheduling heuristics (including both immediate and batch mode scheduling heuristics) based on the remaining allocation times of Cloud resources is proposed. The scheduling heuristics consist of two phases: task ordering, where tasks are ordered prior to execution (when possible), and task mapping, where tasks are mapped to available (unoccupied) Cloud resources. The Cloud scheduling heuristics aim to maximize resource utilization. Scheduling heuristics supported by available information about BoT tasks (e.g., task size) and/or Cloud resource performances are proposed. However, scheduling heuristics that require no information of either Cloud resources or tasks are also proposed.

Also, in this paper, an elastic Cloud resource allocation mechanism that reallocates Cloud resources as needed by the execution of BoTs is proposed. The elastic resource allocation mechanism consists of monitoring the remaining allocation times of Cloud resources as well as their statuses (e.g., busy executing tasks), and determining whether Cloud resources should be reallocated. In addition, the elastic Cloud resource allocation mechanism can dynamically reallocate Cloud resources without requiring information about task completion times or Cloud resources' computing capacities.

Both the Cloud scheduling heuristics and the elastic Cloud resource allocation mechanism were integrated into an agent-based approach for scheduling and executing BoTs in multiple Cloud providers in a concurrent and parallel manner. Agents are endowed with distributed and cooperative problem solving techniques (including the well-known contract net protocol (CNP) [12]) that allow automated and dynamic selection and composition of Cloud resources from a pool of Cloud providers to execute BoTs. In addition, the agent-based problem solving techniques support distributed, concurrent, and parallel scheduling and execution of BoTs in heterogeneous sets of dynamically provisioned Cloud resources allocated in terms of 1-hour time slots.

The significance of this work is that, to the best of the authors' knowledge, it is the earliest work in adopting an agent-based Cloud BoT concurrent scheduling and execution approach endowed with a family of both immediate and batch mode scheduling heuristics adapted to the resource allocation settings (e.g., 1-hour time slots in Amazon EC2 [1]) of Clouds. In addition, it is the earliest work in adopting an elastic Cloud resource allocation mechanism that autonomously and dynamically reallocates Cloud resources to BoT executions. Moreover, the family of Cloud scheduling heuristics supports the dynamic inclusion of new Cloud resources while scheduling and executing BoTs. The contributions of this work are as follows:

(1) Designing and implementing an agent-based testbed for scheduling and execution of BoTs in Cloud environments in a concurrent and parallel manner (Section 2).
(2) Engineering and integrating an elastic Cloud resource allocation mechanism into the agent-based Cloud BoT scheduling approach (Section 3).
(3) Devising and implementing a family of 14 Cloud scheduling heuristics adapted to the resource allocation settings of Clouds (Section 4).
(4) Providing experimental evidence to demonstrate (i) the efficiency of the Cloud scheduling heuristics (Section 5.1), (ii) the effectiveness of the agent-based Cloud BoT concurrent scheduling and execution approach (Section 5.2), and (iii) the efficiency of the elastic Cloud resource allocation mechanism (Section 5.3).

This paper is structured as follows. Section 2 describes the agent-based Cloud architecture for BoT scheduling and execution. Section 3 presents the elastic Cloud resource allocation mechanism. Section 4 includes the definition of the family of 14 Cloud scheduling heuristics. Section 5 presents the evaluation and simulation results of both the family of Cloud scheduling heuristics and of the agent-based Cloud BoT concurrent scheduling and execution approach, as well as the evaluation and simulation results of the elastic Cloud resource allocation mechanism. Section 6 includes a comparison with related work. Finally, Section 7 presents concluding remarks and future research directions.

2. Agent-based Cloud architecture for BoT scheduling and execution

Agents representing consumer, brokers, Cloud providers, and Cloud resources interact among themselves to dynamically select and compose the best (cheapest) available Cloud resources from a pool of heterogeneous Cloud providers to execute and schedule BoTs in a distributed manner. The distributed scheduling and execution of BoTs is supported by an agent-based Cloud architecture (Section 2.1) as well as by a set of agent interaction protocols (Section 2.2) including the well-known CNP.

2.1. Agent-based Cloud architecture

The agent-based Cloud architecture for BoT scheduling (Fig. 1) is composed of: a service ontology, web services, resource agents, service provider agents, broker agents, consumer agents, and a Cloud directory.

(1) The service ontology defines both functional and nonfunctional Cloud resource capabilities. A functional capability defines the functions performed by Cloud resources, e.g., rendering service. A nonfunctional capability defines how Cloud resources execute their functions, e.g., computing capacity.
(2) Web services are interfaces that provide remote access to Cloud resources. Web services are described by the descriptions of the Cloud resources to which they provide access.
(3) Resource agents (RAs) wrap and orchestrate web services. RAs are enlisted in service provider agents.
(4) Service provider agents (SPAs) manage a set of RAs and offer for lease Cloud resources to broker agents. In addition, SPAs map service capabilities enlisted in the service ontology with their RAs' capabilities by performing a one-to-one matching. Moreover, SPAs handle dynamic and elastic Cloud resource reallocation to execute BoTs (see Section 3 for details) by interacting with broker agents.
(5) Broker agents (BAs) offer for lease BoT execution and scheduling services to consumer agents. BAs compose Cloud resources from multiple SPAs, and then schedule for execution consumer agents' BoTs in their corresponding previously composed sets of Cloud resources. In addition, BAs handle dynamic and elastic Cloud resource reallocation to execute BoTs (see Section 3 for details) by interacting with SPAs.
(6) Consumer agents (CAs) submit BoTs to BAs and map BoT tasks to available Cloud resource types by performing a one-to-one matching between tasks' requirements and available Cloud resource types.
(7) The Cloud directory is a listing of BAs' and SPAs' addresses as well as their functional and nonfunctional capabilities. BAs' capabilities are denoted as *broker*, and SPAs' capabilities are described by the functional and nonfunctional capabilities of their enrolled RAs. The Cloud directory is provided by a system agent, to which BAs and SPAs register, and CAs and BAs consult to look for BAs and SPAs, respectively.