Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Agent-based Cloud bag-of-tasks execution

J. Octavio Gutierrez-Garcia[a], Kwang Mong Sim[b,*]

[a] Department of Computer Science, Instituto Tecnológico Autónomo de México, 1 Rio Hondo St., Progreso Tizapan, Mexico City, DF 01080, Mexico
[b] School of Computing, The University of Kent, Chatham Maritime, Kent ME4 4AG, United Kingdom

ARTICLE INFO

Article history:
Received 14 May 2014
Revised 13 February 2015
Accepted 14 February 2015
Available online 20 February 2015

Keywords:
Cloud computing
Multiagent systems
Resource allocation

ABSTRACT

Bag-of-tasks (BoTs) applications are highly parallel, unconnected and unordered tasks. Since BoT executions often require costly investments in computing infrastructures, Clouds offer an economical solution to BoT executions. Cloud BoT executions involve (1) allocating and deallocating heterogeneous resources with possibly different price rates from multiple Cloud providers, (2) distributing BoT execution across multiple, distributed resources, and (3) coordinating self-interested Cloud participants. This paper proposes a novel agent-based Cloud BoT execution tool (CloudAgent) supported by a 4-stage agent-based protocol capable of dynamically coordinating autonomous Cloud participants to concurrently execute BoTs in multiple Clouds in a parallel manner. CloudAgent is endowed with an autonomous agent-based resource provisioning system supported by the contract net protocol to dynamically allocate resources based on hourly cost rates from multiple Cloud providers. In addition, CloudAgent is also equipped with an agent-based resource deallocation system that autonomously and dynamically deallocates resources assigned to BoT executions. Empirical results show that CloudAgent can efficiently handle concurrent BoT executions, bear low BoT execution costs, and effectively scale.

1. Introduction

Bag-of-tasks (BoTs) applications are sets of multiple unconnected tasks, i.e., tasks without ordering constraints, whose execution can be highly parallelized (Carriero and Gelernter, 1989). Examples of BoTs are bioinformatics applications (De Sterck et al., 2005), data-mining applications (Da Silva et al., 2004), image processing applications (Simpson et al., 2007), among others. Due to the frequently high consumption of computing resources and parallel nature of BoTs, BoT execution often requires costly investments in computing infrastructure. Cloud environments emerge as an economical solution to BoT execution, particularly when BoT execution is not frequent enough to justify a high investment in hardware infrastructure. Cloud BoT executions can be highly parallelized because numerous computing resources containing specialized services (such as image rendering applications and bioinformatics applications) can be instantiated and run in parallel to execute BoTs.

Cloud providers offer a heterogeneous pool of parallel, distributed computing resources virtualized and provisioned at several price rates and on predefined timeslots bound by service-level agreements between consumers and providers established through negotiation (Buyya et al., 2009). In addition, price rates of Cloud resources are subject to change according to providers' pricing strategies even for resource types with the same computing specifications. Therefore, Cloud BoT execution in single- and multi-Cloud environments involves the following activities:

(1) Dynamically provisioning (and deallocating) a potentially heterogeneous set of resources from multiple providers offering for lease resources at possibly different price rates.
(2) Coordinating self-interested parties (such as providers and consumers) to establish service-level agreements.
(3) Dynamically distributing BoT execution across resources probably allocated from multiple providers.

As pointed out by Sim (Sim, 2012), selecting providers with different price rates and coordinating task executions in multiple Clouds require resource management systems capable of constantly managing the resource reservation process and making adjustments in response to changing demands. Sim's work on agent-based Cloud computing suggests that agents (Wooldridge, 2009) are appropriate tools for automating these activities because agents are able to make decisions when carrying out tasks on behalf of the users and can interact with other agents.

Cloud BoT execution systems can be categorized into centralized and distributed systems. Centralized Cloud BoT execution systems (Buyya et al., in press; Lee et al., 2009; Vecchiola et al., 2009) rely on a main control entity that manages every other subordinate entity and every process involved in BoT execution, e.g., resource provisioning.

* Corresponding author. Tel.: +44 1634 888943; fax +44 1634 888900.
E-mail addresses: octavio.gutierrez@itam.mx (J.O. Gutierrez-Garcia), prof_sim_2002@yahoo.com (K.M. Sim).

http://dx.doi.org/10.1016/j.jss.2015.02.039
0164-1212/© 2015 Elsevier Inc. All rights reserved.

In distributed Cloud BoT execution systems (Liu et al., 2010), BoT execution results from the interaction among Cloud participants (e.g., providers, brokers, and consumers) potentially programmed with different capabilities.

Existing Cloud BoT execution systems include Aneka (Buyya et al., in press; Vecchiola et al., 2009), Pegasus (Lee et al., 2009), and SwinDeW-C (Liu et al., 2010).

Aneka is composed of multiple worker nodes that are fully controlled by a central master node that dynamically provisions resources from multiple providers based on administrator-defined allocation policies. Aneka's resource deallocation mechanism estimates BoT execution time to elastically deallocate unnecessary resources.

Pegasus is a centralized BoT execution system that provisions and deallocates resources statically from multiple Clouds without taking into account hourly cost rates.

SwinDeW-C is a distributed BoT execution system composed of *coordinator* and *worker* peers that coordinate among themselves to execute BoTs in multi-Cloud environments. Similar to Aneka, SwinDeW-C also has dynamic mechanisms for provisioning and deallocating resources, which are coordinated by high-level peers.

In this paper, CloudAgent, an agent-based Cloud BoT execution tool capable of executing BoTs in single- and multi-Cloud environments is proposed. CloudAgent is composed of autonomous agents acting on behalf of Cloud participants. These agents are designed to interact with one another following the well-known contract net protocol (CNP) (Smith, 1980). The CNP enables agents to dynamically allocate resources based on resources' price rates. In addition, CloudAgent is capable of executing BoTs by concurrently utilizing heterogeneous resources from multiple providers. Furthermore, CloudAgent is also equipped with a distributed deallocation mechanism to dynamically deallocate resources assigned to BoT executions to reduce execution costs.

Cloud consumers using CloudAgent can reduce costs incurred from resource usage by dynamically sampling potentially fluctuating cost rates from multiple providers leasing resources at possibly different price rates. By using CloudAgent, *n* BoTs can be efficiently executed in a concurrent and parallel manner in a multi-Cloud environment while requiring only a linearly-increasing overall number of messages. In addition, by using an individualized deallocation mechanism, CloudAgent achieves cost-efficient BoT executions.

This paper is organized as follows. Section 2 describes the agent-based architecture of CloudAgent. Section 3 presents a 4-stage BoT execution protocol. Section 4 provides empirical evidence to demonstrate the scalability, cost-saving efficiency, and performance efficiency of CloudAgent. Section 5 compares CloudAgent with related work, and Section 6 presents concluding remarks and future work.

2. CloudAgent architecture

CloudAgent's architecture (Fig. 1) consists of consumer agents, broker agents, service provider agents, resource agents, and a Cloud directory.

2.1. Consumer agents

Consumer agents (CAs) are in charge of executing tasks of BoTs on resources by interacting with resource agents. To allocate resources, CAs submit allocation requests to broker agents by adopting the initiator role of the CNP, which consists of sending a *call-for-proposals* message (containing a set of resource definitions to be allocated) to *n* participants (broker agents). From the broker agents' proposals (containing the overall hourly cost of allocating the resources), the CA selects the cheapest proposal, and sends an *accept-proposal* message to the broker agent of the winning bid and *reject-proposal* messages

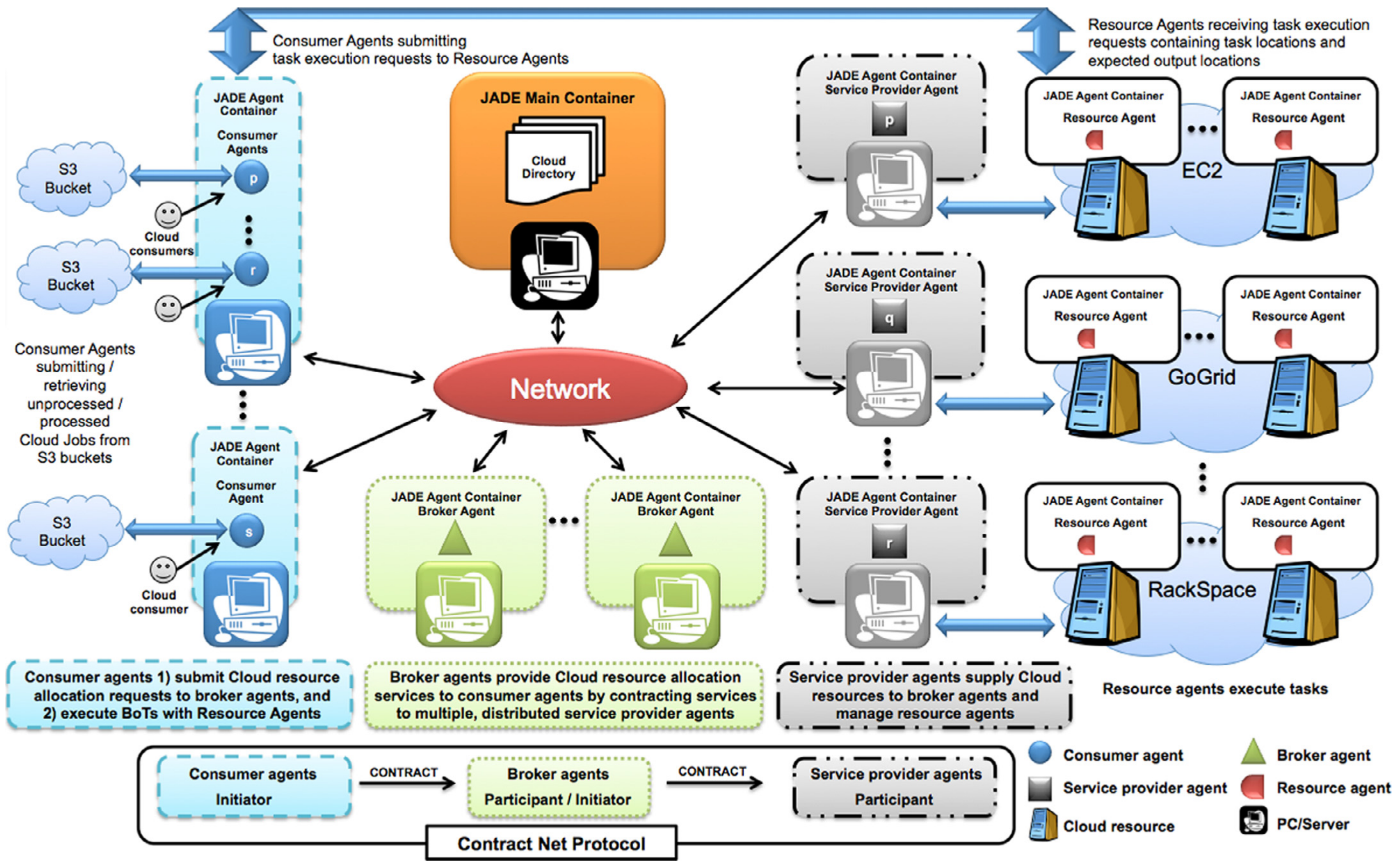
Consumer Agents submitting task execution requests to Resource Agents
Resource Agents receiving task execution requests containing task locations and expected output locations
JADE Agent Container
Consumer Agents
S3 Bucket
Cloud consumers
JADE Main Container
Cloud Directory
Network
JADE Agent Container Service Provider Agent
JADE Agent Container Resource Agent
EC2
GoGrid
RackSpace
Consumer Agents submitting / retrieving unprocessed / processed Cloud Jobs from S3 buckets
JADE Agent Container Consumer Agent
Cloud consumer
JADE Agent Container Broker Agent
Consumer agents 1) submit Cloud resource allocation requests to broker agents, and 2) execute BoTs with Resource Agents
Broker agents provide Cloud resource allocation services to consumer agents by contracting services to multiple, distributed service provider agents
Service provider agents supply Cloud resources to broker agents and manage resource agents
Resource agents execute tasks
Consumer agents Initiator
CONTRACT
Broker agents Participant / Initiator
CONTRACT
Service provider agents Participant
Contract Net Protocol
Consumer agent
Broker agent
Service provider agent
Resource agent
Cloud resource
PC/Server

Fig. 1. CloudAgent architecture.