An Aggregated Utility Function for Negotiation to Model the Preference Ordering of Services in Cloud Computing

Ariya T.K, Christopher Paul, Dr S.Karthik

Abstract— Utility function is generally used to represent an agent's level of satisfaction in terms of price and time. The integration of negotiation with SLA gains much importance. Negotiation activities are needed for establishing contracts and resolving differences between providers and consumers in allocating cloud resources. Intelligent agents providing negotiation support. Controlling the negotiation flow is important for various businesses interested in the SLA. Cloud providers will need to consider and meet different QoS parameters of each individual consumer as negotiated in specific SLAs. This paper explains negotiation technique and simulation of cloud computing systems.

Index Terms— Cloud computing, Price Negotiation, Scientific workflows, Time Negotiation

1 INTRODUCTION

For various businesses, software as a negotiated service is achieving popularity due to technical reasons. This paradigm includes Software as a Service (SaaS), Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). In traditional system, software purchased according to a license. The process of negotiation depends on an agreement between the service customer and the service provider. In service oriented computing flexibility become necessary for an open market.

Negotiation protocols which determine the rules and they are used for collecting negotiating parties. Various styles such as single round to multi round negotiations have been observed. To maximize a utility function SLA is evaluated. In a market, service advertisement done by provider and based on the interest customer's shortlist providers. The provider provisions the agreed upon resources and the customer starts to use the service from the time SLA comes into effect [10].

Cloud computing is an internet based computing solution which provides shared resources. On demand Allocation of resources is flexibility of cloud computing. Grid computing and utility computing combined together in cloud computing. Cloud computing is a way of computing where service is provided across the internet using the models and levels of abstraction [1][11].Grid computing is sharing of coordinated resources in a dynamic environment in which multi-institutional organizations involved.

An agent negotiates over both price and time in a distributed negotiation mechanism. Amazon's Elastic compute cloud used for combinatorial auctions and the fixed price models for achieve high social welfare [12]. A very challenging task for the consumers is simultaneously access several resources according to both providers and consumers needs [2].

For allocating resources according to the customers and providers needs require negotiation activities.

2 Background

There are various negotiations which can occur in cloud computing. Based on the resource allocation policies various negotiation techniques can be used such as workflow level and task level[14]. This section discusses closely related works on Grid resource negotiation, concurrent negotiation, meta negotiation and SLA negotiation.

2.1 Grid Resource Negotiation:

A two-phase bargaining protocol used for Grid resource negotiation [3]. The negotiation protocol consists of

2.1.1 A distributive negotiation phase, in which self-interested agents adopt heuristic strategies to iteratively exchange bids (make proposals and counterproposals) among themselves.

2.1.2 An integrative negotiation phase, in which agents attempt to find joint gains while trying to maintain the utility distribution outcomes from the distributive negotiation phase.

2.2 Concurrent Negotiation

One-to-many negotiation model consists of one buyer and multiple sellers, and the buyer has a number of sub negotiators [4]. There are multiple negotiation threads, and in each negotiation thread, each different sub negotiator conducts a one-to-one negotiation with a different seller.

2.3 Meta Negotiation

A service provider publishes descriptions and conditions of supported negotiation protocols into the registry. Service con-

sumers perform lookup on the registry database by submitting their own documents describing the negotiations that they are looking for [5]. The registry discovers service providers who support the negotiation processes that a consumer is interested in and returns the documents published by the service providers. After an appropriate service provider and a negotiation protocol is selected by a consumer, negotiations between them may start according to the conditions specified in the provider's document [13].

2.4 SLA Negotiation

Preliminary requirements, which are relevant to support negotiation activities into the Cloud, is the definition of QoS parameters for existing service. QoS parameters are necessary to fill the services request in order to negotiate the Cloud, describe the Cloud offer, match the compliant services and build the best available solution, define the SLA and monitor the service levels [6].

SLA negotiation with many Cloud providers for searching for available Cloud services, compliant with user needs, checking trustiness of providers, deciding with whom to negotiate, negotiates the best price for the same offer by different provider, negotiating of multiple SLAs with different providers and to overcome the lack of one compliant offer by a single provider [15].

Cloud service consumer deals with business objectives, preferences and requirements. At the same time service providers default terms and restrictions [16]. Then both consumer broker and provider broker negotiate each other.

3 Scientific Workflow

Scientific workflows in domains such as high-energy physics and life sciences utilize distributed resources in order to access, manage and process large amount of data from a higher-level. in the cloud stack are Software-as-a-Service providers who offer end-users with standardized software solutions that could be integrated into existing workflows. We start by reviewing existing solutions for workflow applications and their limitations with respect to scalability and on-demand access [7]. This enables workflow management systems to readily meet Quality-of-Service (QoS) requirements of applications, does not need advance reservation of resources in global multi-user Grid environments.

3.1 Workflow Design

Workflow design finds how workflow components can be defined and composed.

3.1.1 Workflow Structure

A high-level architectural view of a Workflow Management System (WfMS) utilizing cloud resources to drive the execution of a scientific workflow application. For instance, a policy for scheduling an application workflow at minimum execution cost would utilize local resources and then augment them with cheaper cloud resources if needed, than using high-end but more expensive Cloud resources [8].

A policy for scheduled workflows to achieve minimum execution time would always use high-end cluster and Cloud resources, irrespective of costs.

3.1.2 Workflow Model

Workflow model is also called as Workflow specification defines a workflow including its task definition and structure definition. It consists of two types namely, abstract and concrete. In the abstract model[18], a workflow is described in an abstract form.

3.1.3 Workflow Composition

The objectives for modeling and executing a workflow on Clouds are design an execution model expressed in the form of a workflow, such that multiple distributed resources can be utilized. Parallelize the execution of tasks for reducing the total completion time[9].

Dynamically provision compute resources needed for timely completion of the application when the number of tasks increases. Repeatedly carry out similar experiments as and when required. Manage application execution, handle faults, and store the final results for analysis.

When more tasks began completing as a result of adding new resources, the workflow engine was able to submit additional tasks for execution.

As demonstrated in this document, the numbering for sections upper case Arabic numerals, then upper case Arabic numerals, separated by periods. Initial paragraphs after the section title are not indented. Only the initial, introductory paragraph has a drop cap.

4 CITATIONS

4 MODULE DESCRIPTION

4.1 Developing the Price Utility Function

In this module develop a price utility function. Normally utility function $U(x)$ represents an agent's level of satisfaction for a negotiation outcome x. Since each Cloud participant has different preferences for different prices and time slots, a price utility function, a time-slot utility function, and an aggregated utility function are used to model the preference ordering of each proposal and each negotiation outcome. Price Utility Function whereas consumers prefer the cheapest price for leasing a service, providers want to sell their services at the highest prices.

Let IPC and RPC (respectively, IPP and RPP) be the most preferred (initial) price and the least preferred (reserve) price of a consumer (respectively, provider) agent. Let P be a price that both agents reach an agreement. Min is the minimum utility that a consumer and a provider receive for reaching a deal at their respective reserve prices. To differentiate between not reaching an agreement and reaching an agreement at the reserve price, uP min is defined as 0.01. If a consumer or a provider cannot reach an agreement before its negotiation deadline, both agents receive a utility of zero since not reaching an agreement is the worst possible outcome. The range of the consumer's and provider's price utility functions is {0}∪ [uP min, 1].

4.2 Implementation of time slot utility function

In this module, a novel time-slot utility function is designed to model consumers' and providers' preferences for different time slots. In general, a consumer can have multiple sets of acceptable time-slot preferences A provider's time-slot preferences are based on the following: 1) Service demand (it is more difficult to schedule jobs at a time when the demand is high); 2) temporal ordering (scheduling jobs at the earliest possible time is preferred because computing resources devaluate with time); and 3) fitting job size (to optimize resource utilization). Three formulas are used to characterize service demand, temporal ordering, and fitting job size.

A function consisting of a weighted combination of these three formulas is used to prioritize all the available time slots. Then, a mapping function assigns a priority value to each time slot. Finally, the time-slot utility function transforms the priority value of a time slot into a number from 0 to 1

4.3 Implementing the Consumer time-slot utility function

In this module a consumer generally describes different preferences for multiple sets of time slots. The time-slot utility function of a consumer consists of partial functions for modeling preferences for different time slots. A consumer can select multiple reference points. Each reference point Txm is assigned a utility value Uxm to represent the time-slot preference. A reference point is used to generate the xth partial function of the time-slot utility function.

Each time slot is associated with a utility value, and the time-slot utility values are assigned between uT min and 1 according to the preference priority. uT min is the minimum utility that the consumer receives for reaching a deal at its worst (or least preferred) time slot within FTC and LTC—the range of available time slots for a consumer. FTC and LTC are the indices of the first and last time slots selected by the consumer, respectively. For the purpose of experimentation, uT min is defined as 0.01. A consumer receives a utility of zero if it cannot reach an agreement with the provider on a mutually acceptable time slot before its negotiation deadline

4.4 Implementation of Provider's time-slot utility function

In this module the providers may prefer to allocate jobs to time slots where a low service demand (or resource load) is expected (when there are many simultaneous requests, it is harder to schedule jobs because of limited resource capacities); to their earliest available time slots (since computing resources devaluate with time, unused resources lead to loss of revenues for the providers); and to the time slots at which the job sizes can be accommodated to optimize resource utilization.

A provider prioritizes its time-slot preferences based on expected service demands, and the preference for each time index T is associated with a time-slot priority VD (T). FTP and LTP are the indices of the first and last time slots selected by a provider, respectively. Based on the demand pattern, a provider can assign a lower time-slot priority (i.e., low VD (T)) to the peak time since providing services at a time slot with a lower priority can be compensated by charging a higher price. The provider can assign a higher priority (i.e., high VD (T)) to time slots when low service demands are expected and charge a lower price for these off-peak time slots to provide incentives for consumers to run their applications at time slots with low service demands.

4.5 Implementing the negotiation strategy

In this module negotiation is takes place on both price and time slot, generating a counterproposal can be making either a concession or a tradeoff between price and time slot. Hence, an agent's strategy for multi-issue negotiation is implemented using both the following: 1) a tradeoff algorithm and 2) a concession making algorithm.

New Tradeoff Algorithm called a "burst mode" proposal, which is designed to enhance both the negotiation speed and the aggregated utility. In the burst mode, agents are allowed to concurrently make multiple proposals, with each proposal consisting of a different pair of price and time slot that generates the same aggregated utility. These concurrent proposals differ from each other only in terms of the individual price and time-slot utilities

The concession-making algorithm determines the amount of concession total for each negotiation round, which corresponds to the reduction in an agent's expected total utility. Agents in this work adopt the time-dependent strategies to determine the amount of concession required for the next proposal.

4.6 Decision-making process

In this module resources negotiation developed between Cloud Coordinators. The party that sends the message is defined the operation type, whereas decision is made by the party that receives the message. Therefore, BUY means that the sender of message wants to buy resources and SELL means that the sender wants to sell resources. Offer type is defined by the sender based in Alternate Offers operations. Data: offer: Alternate Offers message received from a remote Cloud Coordinator. Data: required Resource: description of resource under negotiation, defined in the initiate Offer message received in the beginning of negotiation.

4.7 Performance Measure

To evaluate the performance of the burst mode using the PTN mechanism, we used the following as the performance measures: 1) negotiation speed and 2) average total utility of the negotiating pair. The negotiation speed *is* a function of the number of negotiation rounds spent on negotiation. Average total utility is the level of satisfaction in terms of price and time slot with the service to be provided.

Algorithm 1 Decision-making process

1 Get resourceValue
2 if offer.type = SUBMIT then
if(offer.operation=BUY^ offer.value>resourceValue)
∇(offer.operation = SELL ^
offer.value< resourceValue) then /* good offer: accept it. */
4 send ACCEPT;
5 end
6 else not a very good offer: try to negotiate.
7 send COUNTER(resourceValue);
8 end
9 end
10 else if offer.type = COUNTER then
11 if (offer.operation = BUY ^ (offer.value
<resourceValue ^offer.value > lastOffer))
_(offer.operation = SELL ^ (offer.value > resourceValue ^ offer.value < lastOffer))
Then party upgraded its offer, but it is still not good enough: try to negotiate.
12 send COUNTER(resourceValue);
13 end
14 else
party did not upgrade its offer then reject.
15 send REJECT;
16 clean(requiredResource);
17 end
18 end
19 else
Party rejects our counter: finish process.
20 clean (requiredResource);
21 end

5 Conclusion

We have presented negotiation techniques is capable to prevent the loss in price for both providers and customers. The final negotiation result will be either the compromised QoS requirements or a failed submission of the cloud workflow instance. A Scientific Workflow System provides mechanism to gracefully handle the resource negotiation. This paper discussed the negotiation techniques in cloud computing and focus on scientific workflow based on price and time utility functions and simulation of cloud computing systems.

References

[1] Edwin, Philipp Wieder,A Generic Platform for Conducting SLA Negotiations
[2] Kwang Mong Sim, Senior Member, IEEE, and Benyun Shi, 2010, Concurrent Negotiation and Coordination for Grid Resource Coallocation, IEEE Transactions on Systems, Man, and Cybernetics: Cybernetics, Vol. 40, No. 3
[3] Seokho Son and Kwang Mong Sim, 2012, A Price- and-Time-Slot- Negotiation Mechanism for Cloud Service Reservations, IEEE Transactions On Systems, Man, And Cybernetics—Part B: Cybernetics, Vol. 42, No. 3
[4] Saurabh Kumar, Rajkumar Buyyaa, Howard Jay Siegel, 2010,Time and cost trade-off management for scheduling parallel applications on Utility Grids, Future Generation Computer Systems 26 1344_1355
[5] Bo An, Victor Lesser, David Irwin, 2008, Automated Negotiation with Decommitment for Dynamic Resource Allocation in Cloud Computing4, DOI: 10.1109/ ICCCNET.2008. 4787663 pp. 1-9, IEEE explore
[6] Ivona Brandic, Srikumar Venugopal, Michael Mattess,and Rajkumar Buyya, Towards a Meta Negotiation Architecture for SLA-Aware Grid Services
[7] DivyaJyothi.Madhe, D.R.Ingle, 2012, Dealer Agent based Cloud Ecommerce Framework" International Conference on Advances in Communication and Computing Technologies (ICACACT)
[8] Mario Macıas, J. Oriol Fito and Jordi Guitart, 2010, Rule-based SLA Management for Revenue Maximisation in Cloud Computing Markets, CNSM
[9] Mario Macías, Jordi Guitart,A Genetic Model for Pricing in Cloud Computing Markets
[10] Rabi Prasad Padhy, Dr. Manas Ranjan Patra,Dr. Suresh Chandra Satapathy, 2012 ,SLAs in Cloud Systems:The Business Perspective",InternatIonal Journal ofComputerSCIenCe and technology Vol. 3, ISSue1
[11] Rajkumar Buyya, Chee Shin Yeo and Srikumar Venugopal, Market- Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities
[12] Rajkumar Buyya,Rajiv Ranjan, Rodrigo N.C ,InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services
[13] N.Malarvizhi, Dr.V.Rhymend Uthariaraj. (2009): A Minimum Time to Release Job Scheduling Algorithm in Computational Grid Environment, IEEE Fifth International Joint Conference on INC, IMS, IDC.
[14] Sameer Singh Chauhan, R. C. Joshi, (2010), QoS Guided Heuristic Algorithms for Grid Task Scheduling, International Journal of Computer Applications (0975 – 8887), Volume 2 – No.9
[15] He X, Sun, X., Laszewski, G.V., (2003). Qos guided min-min heuristic for grid task scheduling, Journal of Computer Science and Technology 18, 442-451.
[16] Q. Zheng, B. Veeravalli, and C. Tham.(2007): Fault-tolerant Scheduling for Differentiated Classes of Tasks with Low Replication Cost in Computational Grids, ACM, HPDC"07, June 25–29, 2007, Monterey, California, USA.