Evaluation of a Secure Agent Based Framework for Optimized Resource Scheduling in Cloud Computing

Aarti Singh and Manisha Malhotra
Maharishi Markandeshwar University, Mullana, Haryana, India

Abstract.

This work presents secure agent based framework for optimized resource scheduling in CC (SAFORS). It is an extension of authors earlier work in which different frameworks has been proposed. This work explores the real world applicability by integrating all proposed framework. SAFORS has been implemented and it is evaluated based on metrics important for cloud environment. Results obtained are promising.

Keywords: Cloud computing, Evaluation, Intelligent Agents, Multi Agent System, Optimization, Resource Scheduling.

Introduction

Cloud computing (CC) has gain widespread popularity because of economic and operational benefits offered to IT based organizations or setups. CC offers on demands self-service, ubiquitous network access, location independent resources, pooling, rapid elasticity and measured services [11]. All these features make CC appealing and are attracting individuals and organizations to shift towards it. However, despite all the hypes about CC, well established large organizations are reluctant to shift towards this new computing model, as it seems to end users like a black box. Users are receiving desired services and paying for the same, but they don’t have any control over security and accessibility of their data. This is one of the major concern of users about CC. Additionally, multi tenancy, resource sharing, load balancing and trust management are some other challenges [12] still prevailing in this domain. This work is an extension of authors earlier work, in which individual agent based frameworks for cost optimization of resource provisioning in CC, load balancing algorithm and two tier hybrid security framework has been proposed. This work presents secure agent based framework for optimized resource scheduling in CC (SAFORS), which integrates all earlier proposed frameworks. This mechanism provides layers for optimized resource scheduling, load balancing and security of data in cloud environment. Evaluating this framework on available metrics to check its applicability for cloud environment is the motivation of this paper.

The structure of paper is as follows: Next section presents working of proposed integrated SAFORS framework. Section 3 presents relevant literature and explores parameters suitable for evaluation of this mechanism. Results and implementations will be discussed in section 4 and finally section 5 concludes the whole work and further outlines the future scope in this field.

Secure Agent Based Framework for Optimized Resource Scheduling in CC (SAFORS)

SAFORS is an integrated framework facilitating optimal resource allocation, load balancing and secure communication in cloud environment. It comprises of two layers namely resource allocation layer and security...
Resource allocation layer is responsible for optimal resource allocation with optimized cost. Security layer is responsible for improved data security in CC. High level view of SAFORS is as given below:

SAFORS comprises of various agents, working in its different modules for providing automated service composition (ASC) [6], autonomous load balancing (A2LB) [2] and hybrid two tier security (HT2SE) [5]. Agents present in each framework are responsible for a specific function and task. All of these frameworks are arranged in such a manner so that an agent communicates with other concerned agents only and overall complexity of mechanism is not increased. The detailed view and two layers of SAFORS are shown in figure 2 given below.

Figure 1. High Level View of SAFORS
Description of all agents used in SAFORS is as follows:

- **Assistant Agent (AA):** AA keeps the record of all available VM in local DC [6]. After finding suitable resources, it forwards the request with appropriate resources to resource manager agent (RMA). In case of overloaded VM it forward the request to load agent (LA).

- **Resource Manager Agent (RMA):** Resource Manager Agent receives user requests and corresponding resources recommended by AA. It plays the final and important role in SAFORS by allocating the suitable resources to every corresponding request [6].
• **Load Agent (LA):** The major work of a load agent is to calculate the load on every available virtual machine after allocation of a new job in the data centre [2].

• **Channel agent (CHA):** On receiving the request from load agent, the channel agent will initiate some migration agents to other data centers for searching the virtual machines having similar configuration as in concerned data center. It maintains this information for future usage [2].

• **Migration Agent (MA):** These agents are initiated by channel agent. It will move to other data centers and communicate with load agent of that data centre to enquire the status of VMs present there, looking for the desired configuration. On receiving the required information it communicate the same to its parent channel agent. Afterwards, it will stay at destination location, waiting for self-destroy message from parent CA channel agent. The status of migration agent may be alive or destroyed based on its applicability [2].

• **Crypto Agent:** This agent is responsible for encryption and decryption of data at client end. It is equipped with user’s set of keys. Whenever a user gets registered with a CSP, CA exchanges its ECC Private Key with ECC public key of cloud service provider agent. When user sends some data to cloud data center, CA first encrypts it with blowfish algorithm and then with ECC algorithm. In case some data is received from cloud data base it decrypts it with ECC key of CSP [5].

• **Cloud Service Provider Agent (CSPA):** This agent is responsible for interacting with crypto agent of user. It receives encrypted data from the user, places it in cloud data base. It also keeps record of user ECC public key in server key log file. Whenever user requests for its data, it authenticates user, then it decrypts data to level1 using ECC public key of concerned user and then again encrypts that data using ECC private key of CSP. Then it sends data to the user. CSPA knows blowfish key of service provider and uses it for encrypting its Key server log so that even if a hacker hacks this database, it can’t access user data [5].

The detailed description of layers and working of SAFORS framework is discussed in next subsection:

**Allocation Layer**

This layer is responsible for two major tasks:

- Optimized Resource Scheduling
- Load Balancing

On every incoming requests, whether it is Saas, Iaas or Paas, assistant_agent (AA) gets activated and start finding all potential and available resources in its database. While searching, there are two options, either the resources are available in its own data center (DC) or they are not. If resources are found by AA even then it is possible that concerned virtual machine (VM) is underloaded or overloaded. Equation 1 represents the status of DC. In case of underloaded VM, its status would be 0 and for overloaded VM it would be 1.

\[
DC_{vm\_status} = \begin{cases} 
0 & \text{Under Loaded} \\
1 & \text{Over Loaded} 
\end{cases}
\]  

(1)

**Case I:** \(DC_{vm\_status} = 0\)

One AA keeps the record of a DC and status of every VM available in it. After finding the suitable VM, AA forwards the information to resource_manager_agent (RMA) which applies the automated request processing mechanism for optimizing resource allocation [6]. AA search appropriate instances of VM and forwards the list of specification to resource_manager_agent (RMA). RMA has an important and final role in the proposed framework means to allocate the resources recommended by AA against the request forwarded by user. RMA also update the resource_repository and allocation table after completion of a task.

**Case II:** \(DC_{vm\_status} = 1\)
In this case VM of a DC are overloaded, then possible solution is to adapt scalability [4]. Load agent is responsible to ensure its scalability. It works on autonomous agent based load balancing algorithm (A2LB) [2]. It works proactively for calculating load status of various VMs available in its DC. LA periodically determines the workload of virtual machines in terms of available memory, available CPU utilization, and expected response time. Afterwards it calculates the fitness value of each virtual machine which is directly proportional to the memory of a machine and is calculated by equation 2, 3 and 4:

$$\mu_{available} = \mu_{total} - \mu_{used} \ (2)$$

$$\nu(\%) = \frac{\mu_{available}}{\mu_{total}} \times 100 \ (3)$$

$\mu$ is the percentage of memory used and $\nu$ is the fitness value for a virtual machine.

The percentage of fitness values gives the status of a virtual machine.

$$\nu = \begin{cases} 
\leq 25 \% & \text{Critical Allocation} \\
> 25 \% & \text{Normal Allocation}
\end{cases} \ (4)$$

Load agent calculates percentage of $\mu$ and $\nu$ since these factors affect processing of incoming requests. Based on value of $\mu$ available, fitness value ($\nu$) for each node is generated. As long as $\nu$ of a node is greater than the threshold value i.e. 25%, in this case VM status is normal. As and when fitness value of a VM becomes less than or equal to threshold value, load balancing needs to be performed. Load agent on observing critical status of a VM will intimate and send the specification of that VM to the channel agent (CHA). The channel agent will initiate the migration agents to other data centers for searching the virtual machines having similar specifications. Migration agents (MA) being ants will travel only one way. On reaching a destination data centre, MA will first send an acknowledgement message to its parent CHA. Afterwards it will check with load agent of that data centre for availability of virtual machines having similar configuration as desired. If no such VM exists at that data centre, migration agent sends a <Not-Applicable> message back to its parent CHA and waits for <self_destroy> instruction from it. However, if one or more VMs having desired configuration are found, MA further checks their $\mu$ and $\nu$ status and sends this information to CHA. LA forwards the suitable resources to AA, further AA passes this information of requests and corresponding resources to RMA for allocation.

**Security Layer**

Data security is of utmost important in CC. Since all user data is kept on cloud server, it has to be secured at cloud as well as in the network. Security layer of SAFORS uses hybrid two tier security engine based on HT2SE [5] mechanism. This mechanism has two layers, first layer makes use of symmetric key algorithm i.e. Blowfish to encrypt data, this key would only be known to the user. Output of the first layer would be processed by second layer which would again encrypt same data with asymmetric key ECC, for this layer ECC private key will be with user and corresponding public key will be with CSP. After user registration, whenever data is received from the cloud user for
storage on cloud database, crypto agent (CA) gets activated. It first applies Blowfish algorithm for converting original data into cipher form in level 1 and then it applies ECC private key to encrypt cipher text obtained from level 1 to encrypt it for the second time. It then sends this double encrypted data for travelling in the network and further storage in cloud database. When data reaches cloud data center, CSPA gets activated, it fetches user ECC public key stored in Server key log and decrypts data to level1 and then saves it in cloud data center’s data base. Whenever data is required by a particular user, CSPA fetches requested data, encrypts it for the second time using its own ECC private key and sends it on the network. This way data is always double protected while travelling in the network and both encryption keys can’t be hacked as they are distributed.

Next section discusses the possible evaluation parameters of SAFORS.

**Evaluation Parameters**

A critical review of existing literature reveals that there are no well-defined or standard metrics for evaluating agent based frameworks in cloud computing. Even there is no literature in which authors have evaluated their work in cloud environment. For illustration, Niloofar et al. [11] explained some parameters which effect performance of cloud computing environment. According to them, performance is directly proportional to all factor related to service provider and end users. Finally they highlighted some important factors in two categories. First category comprises of security, scalability, workload, latency and network bandwidth that affects the performance and second category comprises of number of data centers, response time of data center, VM cost and total cost of data center that effect the simulation environment. Repschläger et al. [12] indicated that the required criterion to adopt a cloud solution is scalability, service response time, interoperability, reliability, cost, throughput, efficiency and availability. Beside these parameters of cloud, some author have evaluated their agent based work in other fields like fuzzy logic, semantic web etc. Juneja et al. [7] evaluated their agent based framework in cellular network. They have considered nonfunctional issues like performance, scalability and security. Singh et al. [1] evaluated their work in semantic web. Their work is based on multi agent system and proposed multi agent framework for semantic web (MAFSW). Authors evaluated this mechanism on the basis of performance, scalability, stability, trustworthiness and security. After analyzing parameters affecting the performance of cloud environment, scalability, response time, security, workload and VM cost are chosen as suitable parameters for evaluating performance of our proposed system. Brief description of these parameters is as below:

- **Scalability**: Scalability is the ability of system to accommodate the request either by scaling in or scaling out services. Scale in and scale out service time will affect the performance of system.
- **Response Time**: It is an important parameter that directly affects the performance of a cloud. A cloud service provider (CSP) delivering the services in minimum time will be chosen by users.
- **Security**: Security is of utmost importance to cloud users. CSP must ensure stringent security policies at all levels.
- **Workload**: Workload is defined as the load shared by multiple users on different VMs. Load on various VMs in a data center has to be balanced for optimal utilization of resources.
- **VM Cost**: This is the cost that user has to pay for accessing services. From user’s point of view, it is significant parameter. CSP has to ensure that while scaling in or scaling out services, VM cost should not be affected.

Next section presents the implementation of SAFORS and discusses the results achieved after implementation.

**Implementation and Results**
After a critical investigation on available evaluation parameters, the proposed integrated framework considered VM cost, scalability, response time, workload and security parameters for evaluation of proposed mechanism.

- **VM Cost:** The evaluation of virtual machine cost is the simplest and practical way to compare the cloud service providers. The key concern for cloud service provider is that user should not bear expense of scalability. Scalability is abstracted from user and user is unaware of it. The work described in [6] proves that cost remains unaffected whether the resources are found in local datacenter or from other datacenters. If we look towards the response time, it takes almost double time to serve a request in case scalability is adopted. This is shown in figure 3.

- **Scalability:** Scalability in cloud computing refers to upscale and downscale of the requirement as and when needed and in case of mobile agents it is directly concerned with the performance by changing number of agents. Here the point of concern is that effect of adopting scalability on the complexity of system. Proposed SAFORS mechanism is scalable [4] since it is combination of different agent based frameworks and agents can replicate themselves at any time to cater increased request. The mechanism proposed in [2, 6] deliver full scalability without any additional cost as already been discussed in earlier parameter.

![Figure 3. Variation of VM Cost & Response Time in Same and Different Data](image)

- **Response Time:** A2LB [2] mechanism uniquely contributes towards response time in CC. Because this mechanism has proactive load calculation of VM in a DC and whenever load of a VM reaches near threshold value, load agent initiates search for a candidate VM from other DCs. In figure 3, response time gets doubled in case of different DC but this flaw has been removed in A2LB. It takes approximately same amount of time whether the resources are found in same DC or other DC due to overloading of VM as shown in figure 4. However, if we search resources in other DC without applying A2LB then response time increases to almost double.

![Figure 4. Illustration of Response Time in Different Cases](image)
Workload: In CC workload is the abstraction of actual work that is going to be performed in an instance or set of instances. In SAFORS, increased workload doesn’t affect performance of the system since it comprises of multiple agents each of which is responsible for performing a specific task. Moreover, agents may be replicated as and when required to cater increased user requests. Thus with increase in number of agents, workload per agent gets reduced. Workload of proposed framework is indicated in below figure 5.

![Workload of SAFORS](image1)

Figure 5. Workload of SAFORS

Security: Security is also one of most important parameter of CC. SAFORS gives promising security by providing dual security mechanisms. The hybrid mechanism adopted for security amalgamates symmetric and asymmetric key algorithms to ensure stringent security of data and distribution of encryption keys. Blowfish and ECC encryption algorithms provide efficient security with smaller key sizes, as compared to other existing algorithm. The SAFORS mechanism uniquely contributes towards data security in cloud environment. A prototype implementation indicated small increase in encryption time, as compared with single encryption algorithms, which seems acceptable considering the level of security offered. Figure 6(a), (b), (c) given below indicates this.

![Encryption and Decryption time using ECC](image2)

Figure 6(a). Encryption and Decryption time using ECC
Conclusion

The adoption of cloud computing in IT industries has raised some key issues which need attention from research community. This paper proposed the SAFORS framework which is an integration of author’s earlier work. Also this study evaluated the proposed SAFORS framework on the basis of five important metrics which should be considered before embracing a cloud service. Result obtained by implementing proposed mechanism seems promising. However, trust in cloud environment has been left as dimension of future research.

References


