Using Agents for Dynamic Components Redeployment and Replication in Distributed Systems

Nadim Obeid and Samih Al-Areqi

Abstract. Availability is one of the important criteria that affect the usefulness and efficiency of a distributed system. It mainly depends on how the components are deployed on the available hosts. In this paper, we present a generic agent-based monitor approach that supports the dynamic component redeployment and replication mechanisms which were presented in Avala and E-Avala. Avala and E-Avala were proposed to improve availability in large and distributed component-based systems via redeployment and replication. By reifying the interaction between the system and components, agents can detect when it is necessary to change the configuration and whether redeployment or replication is more appropriate.

Keywords: Distributed Systems, Agents, Availability, Redeployment.

1 Introduction

Distributed Systems (DS) have to face the problem of disconnected operations. In addition to the fact that the initial deployment architecture may not be very suitable, it is difficult to predict, at design time, the applications which the DS has to deal with. Therefore, finding and maintaining a desirable (e.g. availability) deployment architecture that satisfies a given set of constraints is a challenging problem. This is due to the facts that (1) there are many parameters which influence the selection of an appropriate deployment architecture (2) the space of possible architectures is large and (3) there may be constant need to change locations of components to meet changing requirements. This leads to some problems such as availability, dependency management [15]], and dynamic configuration. Hence,

Nadim Obeid · Samih Al-Areqi
Department of Computer Information Systems,
King Abdullah II School for Information Technology, The University of Jordan
e-mail: obein@ju.edu.jo

M. Ali et al. (Eds.): *Contemporary Challenges & Solutions in Applied AI*, SCI 489, pp. 19–25.
DOI: 10.1007/978-3-319-00651-2_3

mechanisms such as components replication and redeployment may be necessary in order to improve availability and reliability [1, 2, 10, 12, 5].

In this paper, we present a generic agent-based monitor approach that supports the dynamic component redeployment and replication mechanisms which were presented in Avala [12] and E-Avala [3]. E-Avala improves on Avala by (1) considering positive and negative dependencies among components and (2) implementing replication taking into consideration negative dependencies. By reifying the interaction between the system and application components, agents can detect when it is necessary to change the configuration and whether redeployment or replication is more appropriate.

In section 2 we discuss Avala and E-Avala. In section 3 we present the agent-based redeployment approach. In Section 4 we discuss previous approaches to replication and redeployment.

2 Avala and E-Avala

In this section we give a brief presentation of Avala [6] and E-Avala [2]. Let h_1, h_2, …, h_k ($1 \le k$) stands for hosts, *MEM(h_i) be the memory of* h_i. C_1, …, C_n ($1 \le n$) stands for components, *MEM(C_i) be the memory of* Ci and FREQ(Ci, Cj) be the frequency between components C_i and C_j. The Avala algorithm [6] starts by ranking all hardware nodes and software components as follows:

$$\mathrm{IHR_i} = \sum_{j=1}^{k} REL(h_i, h_j) + MEM(h_i) \tag{1}$$

The ranking of software components is performed as follows:

$$\mathrm{ICRi} = d * \sum_{j=1}^{n} \mathrm{FREQ}(C_i, C_j) + \frac{E}{MEM(C_i)} \tag{2}$$

Where d denotes contributions of host memory and E contributions of event size of interactions between C_i and C_j.

The next software component to be assigned to *h*, is the one with the smallest memory requirement and which would contribute maximally to the availability function if placed on *h.* The Component Rank (CR) is calculated as follows:

$$\mathrm{CR(Ci, h)} = D_1(\mathrm{Ci, h,n}) + D_2(\mathrm{Ci, h}) \tag{3}$$

where $D_1(C_i, h, n) = d * \sum_{j=1}^{n} \mathrm{FREQ}(C_i, MC_j) * REL(h, f(MC_j))$

and $D_2(C_i, h, n) = \frac{E}{MEM(C_i)}$

and MC_j is a shorthand for mapped Cj, $f(MC_j)$ is a function that determines the hosts of mapped components, *REL(h, f(MC_j)* is a function that determines the reliability between selected host *h,* and hosts of mapped components.

Host Rank (HR) is calculated as follows:

$$\mathrm{HR}(h_i) = \sum_{j=1}^{m} REL(h_i, MH(h_j) + MEM(h_i) \tag{4}$$

where *m* is number of hosts that are already selected.

E-Avala [2] employs the notion Depend(C_i, C_j), not present in Avala, as follows:

$$Depend(C_i, C_j) = \begin{cases} 1 & \text{if Ci depends on Cj} \\ -1 & \text{if Cj depends on Ci} \end{cases} \quad (5)$$

Furthermore, E-Avala takes into considration whether or not is a need for data consistancy check regarding a C_i as shown below in (6):

$$Consis(C_i) = \begin{cases} 1 & \text{if Ci does requires data consistency} \\ 0 & \text{Otherwise} \end{cases} \quad (6)$$

Let h be the selected host, *l* is the level of dependency for system configuration, determined by the designer, and *nm* be number of mapped components (i.e., already been assigned to selected hosts), E-Avala uses the same equations of Avala to calculate the intial ranking and distribution. It improves on Avala by employing two additional functions: RCR (resp. Consis-RCR) that compute Replicate Component Rank without (resp. with) consideration for data consistency.

$$RCR(C_i, h, n, nm) = D_3(C_i, h, n) + D_1((C_i, h, nm) \quad (7)$$

where $D_3(C_i, h, n) = \sum_{p=1}^{n} Depend(C_p, C_i) + \frac{l + 2E}{MEM(C_i)}$

$$\text{Consis-RCR}(C_i, h,n,nm) = D_3(C_i, h, n)*(1-Consis(C_i) + D_1(C_i, h, nm) \quad (8)$$

E-Avala makes a comparison between the selected components for redeployment determined by CR (cf. (3)) and those to be replicated determined by RCR (cf. (5)). The selected component will be the one with the highest value of CR and RCR and that satisfies the constraints of memory, *Loc*, and *Colloc* with respect to the current host h and components which are already assigned. This process is be repeated until *h* is saturated. The performance of Avala and E-Avala is discussed in [2].

3 Agent-Based Redeployment

Agents are specialized autonomous problem solving entities that are suitable for problem solving in DS [6, 8, 9, 10, 11, 12]. The use of agents enables us (1) to keep track of the communication cost, (2) to mange dynamic reconfiguration while the system is operational and (3) to choose the better mechanism (e.g., redeployment or repplication) to maintain availability at minimal cost.

Let H_R (resp. H_T) stands for the host of the requesting component, C_R, (resp. target component C_T). We employ two kinds of Agents: (1) Comp-Agent (CPA), which has the required information about its host's components and has the ability to monitor any frequent interactions between a component on its host and components on other hosts and (2) Comm-Agent (CMA), which manages the communications with the other Host's *CMAs*. Let Cor(C_R, C_T) stand for the cost of request between C_R and C_T. When Cor(C_R, C_T) becomes high (e.g., above a certain threshold), CPA of H_T will negotiate with the CPA of H_R (through CMAs of H_T and H_R) in order to agree on one of the following options : (1) redeploying C_R in

H_T, (2) redeploying C_T in H_R, (3) replicating C_R in H_T, (4) replicating C_T in H_R or (5) no change. Assuming $H_R \neq H_T$, $Cor(C_R, C_T)$ can be defined as follows:

$$Cor(C_R, C_T) = freq(C_R, C_T)*eventsize(C_R, C_T)/reliable(H_R, H_T) \quad (9)$$

where $freq(C_R, C_T)$ represents the frequency of interaction between C_R and C_T, $eventsize(C_R, C_T)$ denotes the size of interactions between C_R and C_T, $reliable(H_R, H_T)$ is the reliability between H_R and H_T. Fig. 1 and Fig. 2 show the agents and negotiations algorithms.

```
Agent_algorithm (hosts, comps)
{
A= Component Agent of component CT;
B= Component Agent of component CR;

 If ( Cor(CR,CT)=high))
  If (Agent Negotiation (A) ==false)
{
      Agent Negotiation (A) =true
 Result=Negotiation (A, B)
        If (Result==result1)
          Redeployment (CR)

        If (Result==result2)
            Replication (CR)

        If (Result==result3)
          Redeployment (CT)

        If (Result ==result4)
          Replication (CT)

          If (Result==result5)
              None
}}
```

Fig. 1 Agent Algorithm

```
Result=Negotiation ( Agent A, Agent B)

{
Exchange information (A,B)

If (Redeployment (CT) ==true)
  Result 1=availability (Redeployment (CT))


If (Replication (CT) =true)
  Result 2=availability (Replication (CT))

If (Redeployment (CR)=true)
  Result 3=availability Redeployment (CR))

If (Replication (CR) =true)
  Result 4=availability (Replication (CR))
 Else
    Result 5=current availability
Result=max (result1, result2, result3, result4)
Return result;
}
```

Fig. 2 Agent Negotiation

We have made some improvement on the **DeSi** simulator [6] in order to simulate interactions between any two components on different hosts. We generate a deployment architecture that consists of 10 components, 3 hosts with their software agents and with availability=.8122 distributed as follows:

Host0 = {0,4,7}, Host1 ={2,6,3,8} and Host3 = {0,5,1,9}

the input value are as in Table 1.

Table 1 Input Values

Input Parameter	Value	Input Parameter	Value
Number of Component	10	Min host reliability	0
No. of hosts	3	Max host reliability	1
Min comp memory (in KB)	2	Min comp event size (in KB)	.01
Max comp memory (in KB)	8	Max comp event size (in KB)	10
Min host memory (in KB)	15	Min host bandwidth (in KB/S)	30
Max host memory (in KB)	30	Max host bandwidth (in KB/S)	100
Min comp frequency (in events/s)	0	Level of dependency	3
Max comp frequency (in events/s)	10		

Let C^i_R where $0 \le i \le 9$ and C^j_T where $0 \le j \le 9$ be two operating components and let Rep(i) (resp. Red(i)) denote replicating C^i_R (resp. redploying) on host of C^j_T. To test the viability of the algorithms, we execute several scenarios.

The values, as generated by the simulator (of an E-deployment architecture), which effect the agents' negotiation results, are shown below: frequency values between components and their memory size in Fig. 3, dependency values in Fig. 4 and reliability values between hosts and their memory size in Table 2.

We now consider two Scenarios. In the first, C_4 makes requests frequently to C_6 (cf.Table 3). The result (cf. Fig. 5) is to replicate C_4 as there are many components dependent on it, and it provides better availability. We could not replicate C6 because there is a need for data consistency and it depends on two components in its host. In the second, C_3 makes frequents requests to C_9 (Table 4). The result (cf. Fig. 6) is that either mechanism is possible. Redeploying C_3 will improve availability because it has more interaction and both positive and negation dependency relations with components in the host of C_9.

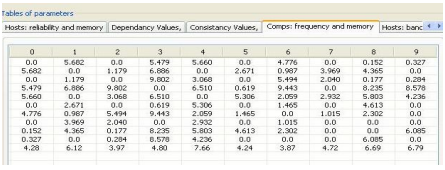

Fig. 3 Components Frequency

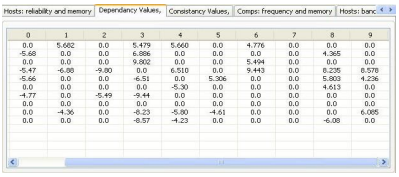

Fig. 4 Component Dependency

Table 2 Host Reliablity/Memory

Host No.	**0**	**1**	**2**
0	1	.49	.38
1	.49	1	.94
3	.38	.94	1
Host MEM	21	27	22

Table 3 Component properties

Comp. properties	**Comp (4)**	**Comp (6)**
Comp. memory size	7,6kb	3,8
Free host size	4,5 kb	12.5 kb
Positive dependency	→0,3	→0,2,3
Data Consistency	0	1

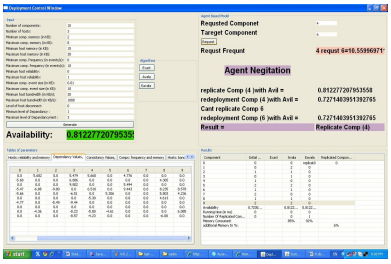

Fig. 5 Senario 1 Results

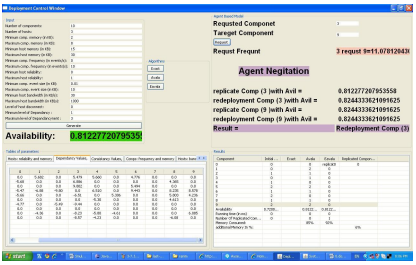

Fig. 6 Senario 2 Results

Table 4 Scenario 1 Component properties

Component Properties	**Component (3)**	**Component (9)**
Component memory size	4.8 kb	6.8
Free Host size	4.5kb	12.5kb
Positive dependency	→0,1,2	→4,5,8
Negative dependency	4,6,8,9→	
Data consistency	0	0

4 Previous Work and Concluding Remark

Several approaches that support the replication of components in DSs have been proposed. However, only a few address redeployment. In [4], Dock is proposed. It employs mobile agents to perform deployment tasks among hosts. It differs from our approach in that it is more concerned with the practical issues of implementing deployment rather than extracting parameters and evaluating deployment architectures. In [5], a constraint-based deployment approach is presented. It addresses the deployment of hierarchical components on heterogeneous dynamic networks. In [3], MARP employs mobile agents to coordinate the updates made to replications maintained at different servers to ensure consistency.

In this paper, we present a generic agent-based monitor approach that supports the dynamic component redeployment and replication mechanisms which were presented in Avala [6] and improved in E-Avala [2]. Some of the issues that need to be addressed include: (1) dealing with functional consistency among components, (2) expanding the solution to include additional parameters such as components structure representation.

References

1. Achmad, I., Graham, M., Santosh, K., Mark, C.: Component Replication in Distributed Systems, A Case Study Using Enterprise Java Beans. In: SRDS, pp. 89–98 (2003)
2. Al-Areqi, S., Hudaib, A., Obeid, N.: Improving Availability in Distributed Component-Based Systems via Replication. In: ACCIDS 2011, pp. 43–52 (2011)
3. Cao, J., Chan, A., Wu, J.: Achieving Replication Consistency Using Cooperating Mobile Agents. In: ICPP Workshops, pp. 453–458 (2001)
4. Hall, R., Heimbigner, D., Wolf, A.: A Cooperative Approach to Support Software Deployment Using the Software Dock. In: ICSE 1999, pp. 174–183 (1999)
5. Hoareau, D., Mahéo, Y.: Constraint-Based Deployment of Distributed Components in a Dynamic Network. In: Grass, W., Sick, B., Waldschmidt, K. (eds.) ARCS 2006. LNCS, vol. 3894, pp. 450–464. Springer, Heidelberg (2006)
6. Mikic-Rakic, M., Malek, S., Medvidovíc, N.: Improving Availability in Large, Distributed Component-Based Systems Via Redeployment. In: Dearle, A., Savani, R. (eds.) CD 2005. LNCS, vol. 3798, pp. 83–98. Springer, Heidelberg (2005)
7. Moubaiddin, A., Obeid, N.: The Role of Dialogue in Remote Diagnostics. In: 20th Int. Conf. on COMADEM (2007)

8. Moubaiddin, A., Obeid, N.: Dialogue and Argumentation in Multi-Agent Diagnosis. In: Nguyen, N.T., Katarzyniak, R. (eds.) New Chall. in Appl. Intel. Tech. SCI, vol. 134, pp. 13–22. Springer, Heidelberg (2008)
9. Moubaiddin, A., Obeid, N.: Partial Information Basis for Agent-Based Collaborative Dialogue. Applied Intelligence 30(2), 142–167 (2009)
10. Moubaiddin, A., Obeid, N.: On Formalizing Social Commitments in Dialogue and Argumentation Models Using Temporal Defeasible Logic. Knowledge and Information Systems (2012), doi:10.1007/s10115-012-0578-6
11. Obeid, N., Moubaiddin, A.: On the Role of Dialogue and Argumentation in Collaborative Problem Solving. In: ISDA, pp. 1202–1208 (2009)
12. Obeid, N., Moubaiddin, A.: Towards a Formal Model of Knowledge Sharing in Complex Systems. In: Szczerbicki, E., Nguyen, N.T. (eds.) Smart Information and Knowledge Management. SCI, vol. 260, pp. 53–82. Springer, Heidelberg (2010)
13. Osrael, J., Froihofer, L., Goeschka, K.: What service replication middleware can learn from object replication middleware. In: MW4SOC, pp. 18–23 (2006)