私立東海大學
資訊工程研究所

碩士論文

指導教授：陳隆彬 博士

對共享資源服務元件尋找最低成本佈署策略

Finding Minimum Cost Deployment Strategy for Service Components with Shared Resources

研究生：周財德

中 華 民 國 99 年 7 月

摘要

由於網際網路服務導向架構的計算模式，使用服務當建立的基本區塊去快速且靈活地發展軟體。為了減少服務管理成本，它是值得的去整合部署那些關係緊密的服務元件部署到相同的主機，基於一定應用具體規則，如全球資源利用策略或接近。本文研究制定一個依據內部和外部應用需求對多個服務應用程式整合部署問題，和一個有效率的演算法發展解決整合部署問題。

關鍵詞：*服務佈署, 服務導向架構, Web 服務*

ABSTRACT

Service-oriented architecture (SOA) is a computing paradigm uses the services as basic building blocks to enable rapid and flexible development of software. In order to reduce the administration cost, it is desirable to integrated deploy tightly related service components based on certain application-specific rules such as global resource utilization policy or proximity. In this paper, we formulate an integrated deployment problem of multiple service applications in terms of intra-application and inter-application requirements. An efficient algorithm is developed to solve the integrated deployment problem.

Keywords: Service deployment, SOA, Web service.

CONTENTS

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

As the conversional application development suffering from flexible software requirements, service-oriented architecture (SOA) adopts the strategy to break monolithic applications into small and reusable components, called *services*. Using services as basic building blocks, SOA enables rapid and flexible development of applications in heterogeneous environments. In the SOA system, a *service provider* is a person or organization that maintains a pool of service instances reside within computer hosts. A *directory service* (*e.g.* UDDI standard [12]) provides a publicly platform for service providers to register their service descriptions (in the form of WSDL [14]) which describe the functionalities of the services formally. By discovering the service directory, an application can locate and bind to the service instances dynamically regardless of their physical locations [15]. Consequently, services can cooperate and interact in a loosely coupled manner and thus enabling flexible composite applications that span organizations.

When building an application comprised of service components, performance could be enhanced if the services are deployed to the computer hosts based on certain rules. For example, two highly interactive services are usually deployed to the hosts proximately. In [2, 17], the previous proposed algorithms solve the *minimum cost deployment problem* for deploying the service components for a single web application. This problem is referred to as the single application deployment problem in this paper. Recently, the researchers [17] showed that the service deployment problem can be solved by using the existing network flow algorithms [1, 3-6, 8, 16, 17].

This paper investigates the problem of deploying N distinct applications over a tree-structured network with (M + 1) hosts. With the previous algorithms, these N applications are deployed individually since the integration among deployment strategies of applications are not considered. This paper proposes the notion of *inter-application* integration which is regarding to deploying different service applications in SOA system integrated. A collection of services (may be used by different applications) but utilizes the same type of resources are deployed in a bundle in order to reduce the administration cost. This paper develops an efficient algorithm that finds the minimum cost deployment strategy satisfying not only the single-application

requirements but also the inter-application integration requirements.

The rest of this paper is organized as follows. In Section II, we review the basic service deployment problem with single application. Section III discusses the integration of multiple application deployments. Section IV discusses our service deployment algorithm based on the properties given in Section III. Finally, conclusions are discussed in Section V.

CHAPTER 2 SERVICE DEPLOYMENT PROBLEM

2.1 Deployment Strategy

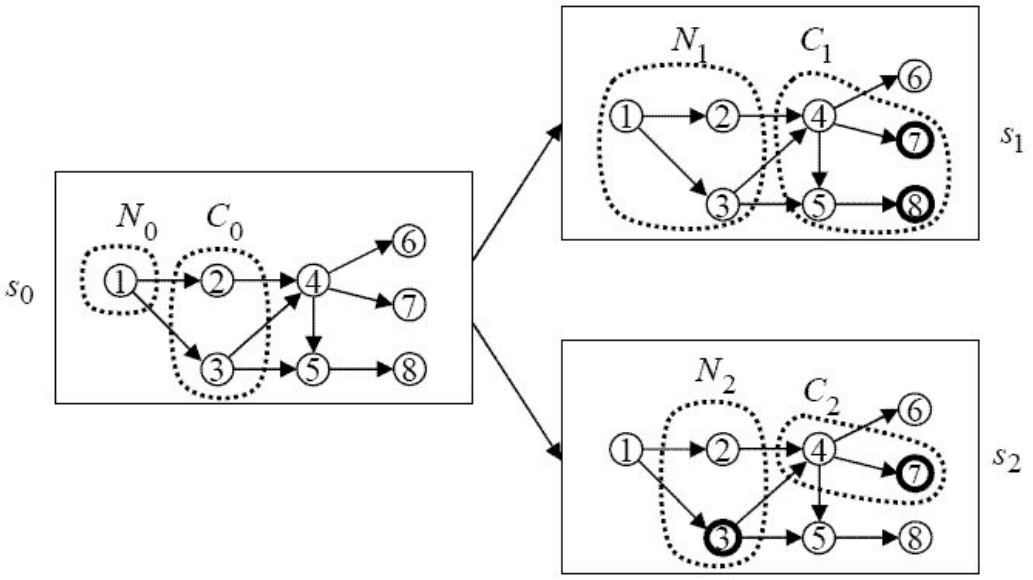


Figure 1. An SDG with 8 services and a collection of deployment items (n_0,c_0) , (n_1, c_1) , (n_2,c_2).

A service application (or *application*) is composed of a set of *service* components which interact based on certain application rules. An application is modeled as directed graph $G = (V , E)$ called SDG (service dependency graph), where each node $a \in V$ refers to a service and each link $(a , b) \in E$ refers to the dependency relation from service a to b. Node (*i.e.* service) a is called a *predecessor* of node b (and b is called a *successor* of a) if edge $(a , b) \in E$. The set of predecessors of node set X ($X \subseteq V$) is given by $pred(X , G) = \{a \mid a \rightarrow b , b \in X , a \in V \}$. Nodes without predecessors (or successor) are called *initial nodes* (or *target nodes*) and are referred to by I (or T). We assume that the dependency relation induced by E is *acyclic*.

The *transcoding* [11, 17] is an important technique to reduce the cost of deploying service components. A transcoding operation is a client-side computation process to construct a service component from the information of its predecessors. The cost of deploying a set of service components can be reduced by taking the trade-off between direct transmitting and client-side computing.

The SOA system consists of $M + 1$ hosts (s_0 , s_1 , . . . , s_M) that provide host environment for managing and executing the service components. The hosts in SOA system are organized in a *tree structure* of which s_0 is the root and s_l ($l > 0$) are client hosts.

In SOA system, *deploying* a service component is to deliver the execution result of the service to a designated computer host. A tuple (n_k , c_k) is called *deployment item* if it is a pair of sets satisfying $n_k \subseteq V$, $c_k \subseteq V$ and $n_k \cap c_k = \phi$. A deployment item defines three possible ways for deploying nodes:

- Nodes in n_k are referred to as *N-nodes* and are deployed to s_k via direct network transmission from s_0 to s_k. That is, an N-node is executed in s_0 and the execution result is then sent to s_k.
- Nodes in c_k are referred to as *C-nodes* and are deployed to s_k via local computation in host s_k. That is, a C-node is executed by client host s_k locally.
- Nodes in $V \setminus (n_k \cap c_k)$ are referred to as *O-nodes*. They are not involved in the application deployment.

An example of SDG and deployment item is shown in Figure 1.

For each node in SDG there are two costs defined on it: the cost of transmitting the execution result of service a from host s_0 to host s_k, denoted by $net(a , s_0 , s_k)$, and the cost of computing a in host s_k, denoted by $comp(a , s_k)$. The cost of deployment item (n_k , c_k) is defined and denoted by

$$cost(n_k , c_k) = \sum_{a \in n_k} net(a , s_0 , s_k) + \sum_{a \in c_k} comp(a , s_k)$$

2.2 Single Application Deployment

A *deployment strategy* consists of a collection of deployment items, one for each host in the SOA system. For simplicity, deployment strategy is also called *deployment* hereinafter. Not every collection of deployment items is feasible. For example, client host s_k can not compute node b

unless all the predecessors of b are available in s_k. There are two ways to make a node x available in client host s_k. First, x is computed by the server s_0 and the execution result is then transmitted to s_k. Alternatively, x can be also computed by s_k when the predecessors are available are s_k.

Definition 2.1 defines the *deployment strategy* for a single application formally.

> *Definition 2.1* (*S-Deployment*): For SDG $G = (V , E)$ and tree-structured network $S =(s_0 , s_1 , . . . , s_M)$, the vector of items $((n_0 , c_0) , (n_1 , c_1) , . . . , (n_M , c_M))$ forms a *deployment strategy* if the following properties hold:
>
> - (Initialization) For root host s_0, $n_0 = I$.
> - (Integrity) For client hosts, $n_l \subseteq (n_0 \cap c_0) , l > 0$.
> - (Consistency) For each host s_k, item (n_k , c_k) satisfies condition $pred(c_k , G) \subseteq (n_k \cup c_k), k \geq 0$.
> - (Fulfillment) All target nodes are required to be available in client host, *i.e.* $T \subseteq (n_l \cup s_l), l > 0$.
>
> The above deployment strategy is called a *S-deployment* for the single SDG G.

In Figure 1 the deployment items form a valid deployment strategy since all the properties in Definition 2.1 are satisfied. The *minimum cost S-deployment problem* is to construct a deployment$((n_0 , c_0) , (n_1 , c_1) , . . . , (n_M , c_M))$ in terms of SDG G and tree network S such that the total cost $\sum_{k=0}^{M} cost(n_k , c_k)$ is minimized.

Assume that an n-node-m-edge SDG G to be deployed over the tree-structured network S with $(M + 1)$ hosts. The minimum cost S-deployment problem in terms of G and S can be solved in time $O(nmM^2 log((nM)^2/mM))$ by reducing the problem to the well-known *maximum flow* problem [2, 4, 6-10, 17].

CHAPTER 3 INTEGRATION OF DEPLOYMENTS OF MULTIPLE APPLICATIONS

The SOA system usually accommodates and operates a large number of applications. Deploying applications individually could cause irregularity of resource distribution and access to the SOA system. In order to reduce the administration cost, it is desirable to deploy a collection of services accessing same resource into the same host. This section defines the problem of integration of multiple application deployments.

3.1 Deployment Matrix

We use the following notations to refer to the SOA system with multiple service applications:

- $\{G^{(1)}, G^{(2)}, \ldots, G^{(N)}\}$ is the set of SDGs each represents an application. For each $1 \le i \le N$, $G^{(i)} = (V^{(i)}, E^{(i)})$. We use $\hat{V}$ to refer to the set of nodes $\bigcup_{1 \le i \le N} V^{(i)}$ in all the SDGs.
- $S = \{s_0, s_1, \ldots, s_M\}$ is the tree-structured network.

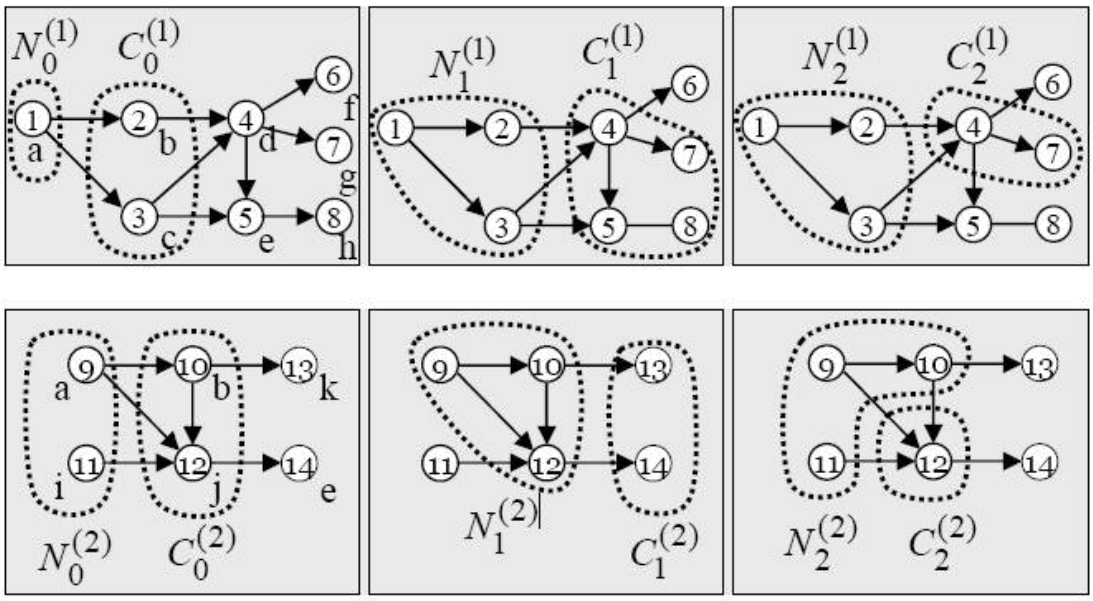
For N applications, there are correspondingly N deployment strategies each consists of $M + 1$ deployment items. The collection of these items forms a matrix as denoted as follows:

- $(n_k^{(i)}, c_k^{(i)})$ is a *deployment item* in terms of application $G^{(i)}$ and host s_k.
- D is an $N \times (M + 1)$ 2-dimensional *deployment matrix* of deployment items $(n_k^{(i)}, c_k^{(i)})$, $1 \le i \le N$ and $0 \le k \le M$.
- The target $T^{(i)}$ is the set of target nodes in application $G^{(i)}$.
- $I^{(i)}$ denotes the initial nodes of application $G^{(i)}$.

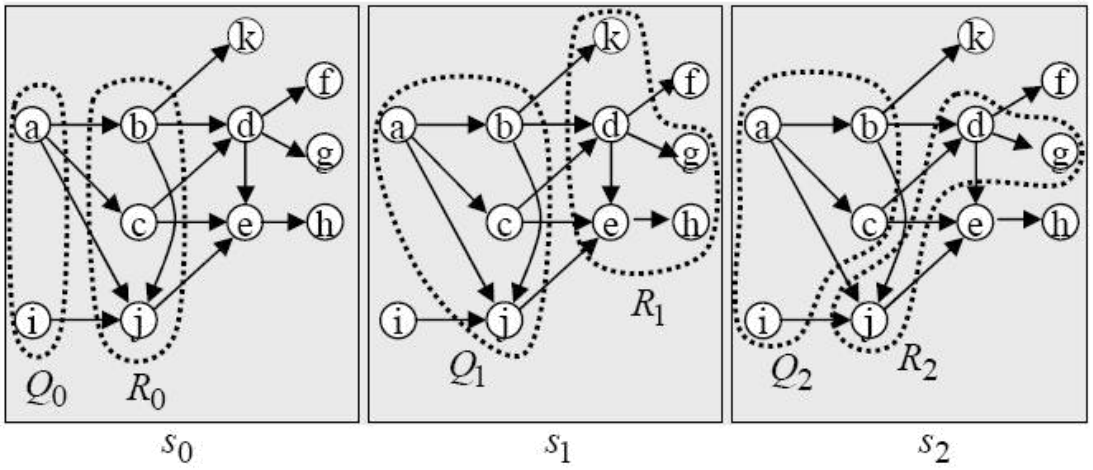
Figure 2 (a) illustrates a 2×3 deployment matrix D for the SOA system with applications $(G^{(1)}, G^{(2)})$ and tree-structured network $S = (s_0, s_1, s_2)$. In this figure, dashed-border areas stands for deployment items.

For example, $((n_0^{(1)}, c_0^{(1)}), (n_1^{(1)}, c_1^{(1)}), (n_2^{(1)}, c_2^{(1)}))$ form a deployment for application $G^{(1)}$. It can be seen that this deployment is

valid as both the consistency and integrity property hold. The integrity property holds since the N-nodes required by client hosts s_1 and s_2 are all available in root host s_0 (*i.e.*, $n_1^{(1)} \subseteq (n_0^{(1)} \cup c_0^{(1)})$ and $n_2^{(1)} \subseteq (n_0^{(1)} \cup c_0^{(1)})$).



(a)



(b)

Figure 2. Deployment matrix of services $G = (G^{(1)}, G^{(2)})$ and network $S = (s_0, s_1, s_2)$: (a) Deployment matrix for $G^{(1)}$ and $G^{(2)}$, and (b) Deployment for $\hat{H}$.

3.2 Integrated Multiple Application Deployment Problem

The SDG only represents *intra-application* dependency relation while lack of correlations among different applications (*e.g.*, views in data warehouses [A27]). In the proposed new problem model, we assign each service (*i.e.* node) in SDG a **type** attribute. A collection of service instances (may be invoked by different applications) requiring a same **type** attribute are deployed as a bundle in order to reduce the administration cost. This paper assumes that each service is associated with only one attribute. Practically, services accessing more than one type of resources are common. This case can be solved by iteratively decomposing the service until it becomes *atomic*, *i.e.* with only one **type** attribute.

Among SDGs, a set of nodes with same **type** attribute are treated as one complex node, called *R-set*, defined in Definition 3.1.

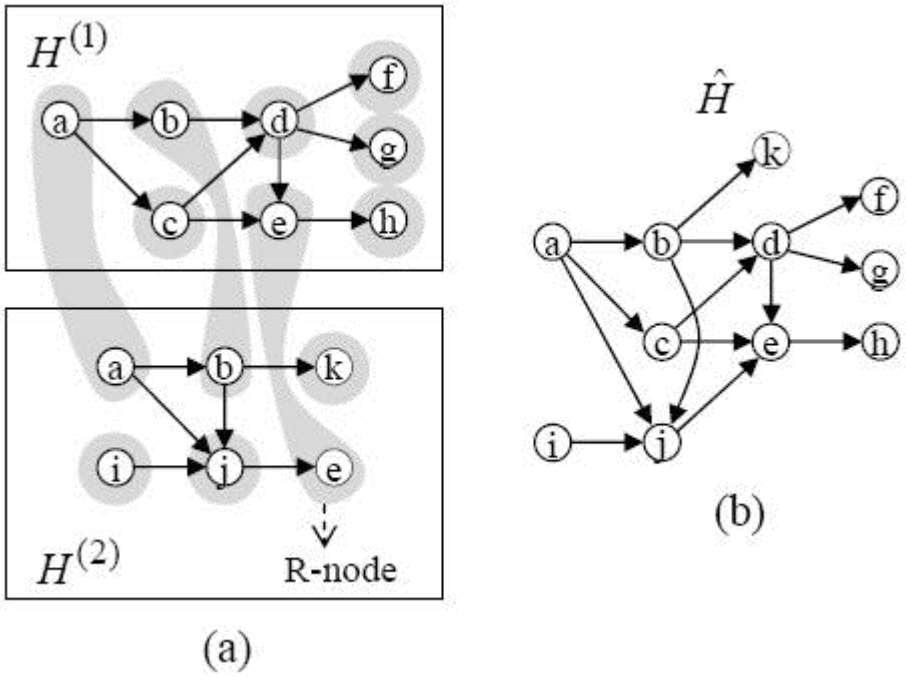


Figure 3. (a) R-graphs $H^{(1)}$ and $H^{(2)}$, and (b) the unified graph $\hat{H}$.

Definition 3.1 (*R-set*): A set of nodes in SDG group ($G^{(1)}$, $G^{(2)}$, . . . , $G^{(N)}$) with same type attributes forms a *R-set*. The collection of all R-sets is denoted by $\Re$. For a node set A (A $\subseteq$ $\hat{V}$), the set of R-sets covering all elements in A is defined as RS(*A*) = $\{\gamma \mid A \cap \gamma \neq \phi, \gamma \in \Re\}$.

Precedence relationship between R-sets γ and γ' is defined based on rule: Relation $\gamma \rightarrow \gamma'$ holds if and only if for any pair of nodes $a \in \gamma$ and $b \in \gamma'$, $(a, b) \in E^{(i)}$ for some *i*. Furthermore, relation $\gamma \rightsquigarrow \gamma'$ is a transitive relation defined by rule: $\gamma \rightsquigarrow \gamma'$ if and only if (1) $\gamma \rightarrow \gamma'$, or (2) $\gamma \rightarrow \gamma''$ and $\gamma'' \rightarrow \gamma'$ for some γ''.

Figure 3 illustrates two SDGs and R-sets among nodes. In this figure, different attributes of nodes are distinguished by different characters, and R-sets are marked by using gray areas. As this figure illustrates, each node with unique **type** attribute forms a singleton R-set. Therefore, the collection $\Re$ forms a mutually exclusive partition on $\hat{V}$ since each vertex in $\hat{V}$ is mapped to one R-set and the mapping is onto.

Deployment of Multiple Applications: Now we are ready to give the formal definition of deployments of multiple applications.

Definition 3.2 (*M-deployment*):

Given application group $G = (G^{(1)}, G^{(2)}, \ldots, G^{(N)})$, and tree-structured network $S = (s_0, s_1, s_2)$, and $N \times (M + 1)$ deployment matrix *D* is an *M-deployment* in terms of *G* and *S*, if the following integration properties hold:

- (Intra-application integration) Each row $D^{(i)}$ is a valid S-deployment in terms of application $G^{(i)}$ and network *S*.
- (Inter-application integration) For each column D_k, the

following property holds for each pair of applications $G^{(i)}$ and $G^{(i')}$:

$$\mathrm{RS}(n_k^{(i)}) \cap \mathrm{RS}(n_k^{(i')}) = \phi$$

In above definition, the *intra-application* integration states that each row $D^{(i)}$ in matrix D refers to a valid deployment for application $G^{(i)}$, while the *inter-application* integration states that the services among different applications are deployed on an R-set basis. In other words, the services with same **type** attribute must be deployed to the same host s_k with same deploying type (N-node or C-node).

An example of deployment matrix with 2 services and 3 hosts is illustrated in Figure 2. It can seen that the strips $\mathrm{D}^{(1)}$ and $\mathrm{D}^{(2)}$ are valid deployments for SDGs $G^{(1)}$ and $G^{(2)}$, respectively. The inter-application integration property can be easily verified for each column strip D_0, D_1, and D_2.

The cost of an M-deployment D is defined as the sum of costs of deploying all the applications:

$$cost(D) = \sum_{0 \le k \le M} \sum_{1 \le i \le N} cost((n_k^{(i)}, c_k^{(i)})) \quad (1)$$

The *minimum cost M-deployment* is a deployment matrix satisfying the inter-application and intra-application integration properties such that the deploying cost is minimized.

CHAPTER 4 INTGRATED MULTIPLE APPLICATION DEPLOYMENT ALGORITHM

The multiple application deployment algorithm is comprised of two main steps: *unifying* and *projecting*. The *unifying* step unifies the group of SDGs based on their R-sets. The unified graph is then treated as normal SDG where the optimal deployment strategy can be determined by invoking the algorithm in Subsection II-C. In second step, this deployment of the unified graph is projected to the original SDGs to obtain the multiple-application deployment strategies.

4.1 Unified Graph

Suppose that $(G^{(1)},G^{(2)}, \ldots,G^{(N)})$ is a SDG group and $\Re$ is the set of R-sets. *R-graphs* are directed acyclic graphs constructed based on R-sets, described as flows. An R-set $\gamma \in \Re$ if and only if node v_γ is in R-graph. In order to distinguish nodes in SDGs and in R-graph, such v_γ is called *R-node*. For each node b in $(G^{(1)},G^{(2)}, \ldots,G^{(N)})$ the R-node corresponding to b is referred to by $rnode(b) = \{v_\gamma \mid b \in \gamma , \gamma \in \Re\}$. Clearly, $rnode(b) = v_\gamma$ if and only if $RS(b) = \gamma$. Definition 4.1 defines R-graph formally.

> *Definition 4.1* (*R-graph*): For SDGs $(G^{(1)},G^{(2)}, \ldots,G^{(N)})$, and their R-sets $\Re$. For each i, $1 \le i \le N$, $H^{(i)} = (U^{(i)} , F^{(i)})$ is the *R-graph* corresponding to $G^{(i)}$ with $U^{(i)} = \{rnode(a) \mid a \in V^{(i)}\}$ and $F^{(i)} = \{(rnode(a) , rnode(b)) \mid (a , b) \in E^{(i)}\}$.

Furthermore, the *unified graph* is a combination of vertices and edges of the collection of R-graphs, as defined as follows.

Definition 4.2 (*Unified graph*): For a set of R-graphs ($H^{(0)}$,$H^{(1)}$, . . . ,$H^{(N)}$), $H^{(i)} = (U^{(i)}, F^{(i)})$, the *unified graph* $\hat{H} = (\hat{U}, \hat{F})$ is defined as $\hat{U} = \bigcup_{1 \le i \le N} U^{(i)}$ and $\hat{F} = \bigcup_{1 \le i \le N} F^{(i)}$.

In unified graph $\hat{H}$, the costs of nodes from different R-graphs are summed up if they are combined in $\hat{H}$. Formally, $\forall x \in \hat{U}$, $net(x, s_0, s_k) = \sum_{1 \le i \le N} \sum_{a \in U^{(i)}, \text{mode(a)=x}} net(a, s_0, s_k)$ and $net(x, s_0, s_k) = \sum_{1 \le i \le N} \sum_{a \in U^{(i)}, \text{mode(a)=x}} net(a, s_0, s_k)$. Figure 3 illustrates a group of SDGs and their unified graph.

Combining graphs could be vague if the graphs come with inconsistent structures. For simplicity, limitations to the unified graph are assumed in this paper:

R1 No cycle is formed in an unified graph $\hat{H}$ when unifying the R-graphs.

Lemma 4.1 shows that $G^{(i)}$ and $H^{(i)}$ are isomorphic.

Lemma 4.1 (*Isomorphism between $G^{(i)}$ and $H^{(i)}$*): An SDG $G^{(i)}$ is isomorphic to its corresponding R-graph $H^{(i)}$.

Proof: The R-graph is constructed based on R-nodes. From the definition of R-node, node *a* is in $G^{(i)}$ if and only if *rnode*(*a*) is in $H^{(i)}$, and the mapping *rnode* is one-to-one. Moreover, edge (*a*, *b*) is in $G^{(i)}$ if and only if edge (*rnode*(*a*) , *rnode*(*b*)) is in $G^{(i)}$. From above, SDG $G^{(i)}$ is isomorphic to R-graph $H^{(i)}$.

Lemma 4.2 shows that unifying multiple SDGs leads to a valid SDG.

Lemma 4.2: The unified graph $\hat{H}$ is a valid SDG.

Proof: This property holds as the unified graph $\hat{H}$ is acyclic (se Property R1) and induces a partial ordered relation on the set of nodes in $\hat{H}$.

4.2 Multiple Application Deployment Algorithm

This subsection discusses the new algorithm for deploying multiple applications in the SOA environment that confirms inter-application and intra-application integration simultaneously. Instead of solving the problem over SDGs ($G^{(1)}$,$G^{(2)}$, . . .,$G^{(N)}$), we solve the problem over the corresponding R-graphs ($H^{(0)}$,$H^{(1)}$, . . . ,$H^{(N)}$). Due to the isomorphic, solutions for the original SDGs can be obtained based on the one-to-one mapping between each pair of $G^{(i)}$ and $H^{(i)}$.

For clarity, deployment strategy for difference groups of SDGs are distinguished via different notations:

- ($n_k^{(i)}$,$c_k^{(i)}$) denote the deployment item of SDG $G^{(i)}$ and host s_k, $1 \le i \le N$ and $0 \le k \le M$.
- ($p_k^{(i)}$,$q_k^{(i)}$)denote the item for SDG $H^{(i)}$ (the R-graph of $G^{(i)}$) and host s_k , $1 \le i \le N$ and $0 \le k \le M$.
- (P_k , Q_k) denote the item for the unified SDG $\hat{H}$ and host s_k , $0 \le k \le M$.

For a set of nodes $\hat{X}$ of $\hat{H} = (\hat{U}, \hat{F})$, define the operation $\hat{X}[i] = \hat{X} \cap U^{(i)}$, where $U^{(i)}$ is the set of nodes in R-graph $H^{(i)}$. For a deployment item (P_k , Q_k) of SDG $\hat{H}$, we can obtain the deployment item ($P_k[i]$, $Q_k[i]$) for R-graph $H^{(i)}$ based on the intersection operation.

Definition 4.3: For unified graph $\hat{H}$ and tree-structed network *S*, the collection of deployment items ($P_k[i]$, $Q_k[i]$) , for all $0 \le k \le M$ and $1 \le i \le N$, forms a $N \times (M + 1)$ *deployment matrix*.

Figure 2 (a) illustrates the deployment matrix that are projected from the deployment of the unified graph $\hat{H}$ shown in Figure 2 (b).

Based on above discussion, the new M-deployment algorithm with input equals to SDGs ($H^{(0)}$,$H^{(1)}$, . . . ,$H^{(N)}$) is described as follows.

Algorithm MINIMUM_M_DEPLOYMENT():

1) Construct the unified graph $\hat{H}$ from the input SDGs ($H^{(0)}$,$H^{(1)}$, . . . ,$H^{(N)}$).
2) Find the minimum cost deployment ((P_0,Q_0), (P_1,Q_1), . . . ,(P_M,Q_M)) of $\hat{H}$ by using the algorithm discussed in Section II-C.
3) Construct a $N \times (M + 1)$ matrix $\hat{D}$ with items ($p_k^{(i)}$, $q_k^{(i)}$), where $p_k^{(i)} = P_k[i]$ and $q_k^{(i)} = Q_k[i]$ for all $0 \le k \le M$ and $1 \le i \le N$.
4) return $\hat{D}$

Since R-graphs $H^{(i)}$ are isomorphic to SDG $G^{(i)}$, the returned deployment matrix $\hat{D}$ (in Step 4) in above algorithm can be mapped to the SDGs based on the one-to-one mapping. The correctness of the above M-deployment algorithm is discussed in next parts.

Correctness and Analysis of the M-Deployment Algorithm: Lemma 4.3 discusses some useful properties between unified graph and the R-graphs comprising the unified graph.

Lemma 4.3: Suppose ($H^{(0)}$,$H^{(1)}$, . . . ,$H^{(N)}$) are SDGs, and $\hat{H}$ is the unified graph unifying them. For two sets of nodes X , Y in $\hat{H}$, the following conditions hold:

1) $(X \cup Y)[i] = X[i] \cup Y[i]$,
2) $pred(X[i] , H^{(i)}) \subseteq pred(X , \hat{H}) \cap U[i]$, and
3) $\bigcup_i \{pred(X[i] , H^{(i)})\} = pred(X , \hat{H})$.

Proof:

(1): $(X \cup Y)[i] = (X \cup Y) \cap U[i]$ and $X[i] \cup Y[i] = (X \cap U[i]) \cup (Y \cap U[i])$, obviously, these two formulas are equal based on the distributive law of sets.

(2): Recall that $\hat{H} = (\hat{U}, \hat{E})$ and $H[i] = (V[i] \cup E[i])$. By definition (in Subsection II-A), the predecessors of set X in $\hat{H}$ is given by $pred(X, \hat{H}) = \{y \mid y \to x, x \in X, y \in \hat{U}\}$, We derive that

$$pred(X, \hat{H}) \cap \mathrm{U}^{(i)} \tag{2}$$

$$= \{y \mid y \to x, x \in X, y \in \hat{U}\} \cap \mathrm{U}^{(i)}$$

$$= \{y \mid y \to x, x \in X, y \in (\hat{U} \cap \mathrm{U}^{(i)})\} \tag{3}$$

The predecessors of set $X[i]$ in i-th R-graph $H^{(i)}$ is given by $pred(X[i], H^{(i)}) = \{y \mid y \to x, x \in X[i], y \in \mathrm{U}^{(i)}\}$. Since $\mathrm{X}[i] = (X \cap U^{(i)})$ and $(U^{(i)} \cap \hat{U}) = U^{(i)}$, this formula $pred(X[i], H^{(i)})$ can be rewritten as

$$pred(X[i], H^{(i)}) = \{y \mid y \to x, x \in (X \cap \mathrm{U}^{(i)}), y \in (\hat{U} \cap U^{(i)})\} \tag{4}$$

Clearly, the set in Equation 3 is a subset of the set in Equation 4. This confirms that $pred(X \cap \mathrm{U}^{(i)}, H^{(i)}) \subseteq pred(X \cap \hat{H}) \cap \mathrm{U}^{(i)}$ *i.e.* the second condition holds.

(3): Decompose the items in the third condition by

$$\bigcup_i [pred(X[i], H^{(i)})] =$$

$$\bigcup_i \{y \mid y \to x, x \in X[i], y \in U^{(i)}\}$$

and

$$pred(X, \hat{H}) = \{y \mid y \to x, x \in X, y \in \hat{U}\}$$

Since $\bigcup_i U^{(i)} = \hat{U}$ and $\bigcup_i X[i] = X$, the above two items are equal and thus implies the third condition.

For convenient, let

$$\hat{P} = ((P_0,Q_0), (P_1,Q_1), \ldots, (P_M,Q_M))$$

and $\hat{D}$ be the matrix of items ($P_k[i]$, $Q_k[i]$) projected from $\hat{P}$ for all $0 \leq k \leq M$ and $1 \leq i \leq N$. Also, $\hat{P}[i]=\{P_k[i] , Q_k[i] \mid 0 \leq k \leq M \}$ refers to row *i* of matrix $\hat{D}$. Theorem 4.1 and Lemma 4.4 prove the correctness of the above M-deployment algorithm.

Theorem 4.1: For a tree-structured network *S*, deployment $\hat{P}$ exists for $\hat{H}$ if and only if the M-deployment $\hat{D}$ exists for SDGs $(H^{(0)},H^{(1)}, \ldots, H^{(N)})$.

Proof: The matrix $\hat{D}$ represents an M-deployment of SDGs $(H^{(0)},H^{(1)}, \ldots, H^{(N)})$ which satisfies both the inter-application and intra-application properties, as proved as follows.

Intra-Application Integration: Given that the deployment $\hat{P}$ satisfies the properties *consistency*, *integrity*, *initialization*, and *fulfillment* for $\hat{H}$, we prove that each *i*-th row of matrix $\hat{D}$ satisfies these four properties for application $H^{(i)}$.

(Consistency) By assumption, $\hat{P}$ satisfies the consistency property, *i.e.* $pred(Q_k, \hat{H}) \subseteq (P_k \cup Q_k)$. Taking intersection on both sides obtains

$$pred(Q_k , \hat{H}) \cap U^{(i)} \subseteq (P_k \cup Q_k) \cap U^{(i)} \quad (5)$$

By Lemma 4.3 (the second condition, letting $\hat{X} = Q_k$), we derive that

$$pred(Q_k[i] , H^{(i)}) \subseteq pred(Q_k , \hat{H}) \cap U^{(i)} \quad (6)$$

Combining Equation 5 and 6 obtains $pred(Q_k[i], H^{(i)}) \subseteq (P_k \cup Q_k) \cap U^{(i)} = P_k[i] \cup Q_k[i]$. Thus, $\hat{P}[i]$ satisfies the consistency property for SDG $H^{(i)}$.

(Integrity) The integrity property is satisfied for $\hat{P}$, thus, $P_l \subseteq (P_0 \cup Q_0)$. Taking intersection on both sides we derive that

$$P_l \subseteq (P_0 \cup Q_0)$$

$$\Leftrightarrow P_l \cap U^{(i)} \subseteq (P_0 \cup Q_0) \cap U^{(i)} \quad (7)$$

$$\Leftrightarrow P_l[i] \subseteq (P_0[i] \cup Q_0[i]) \quad (8)$$

which implies that $\hat{P}[i]$ satisfies the integrity property.

The proof of initialization and fulfillment properties are ignored in this paper. From above, $\hat{P}[i]$ satisfies the consistency, integrity, initialization, and fulfillment properties and thus is a valid deployment for SDG $H^{(i)}$.

Inter-Application Integration: Since $\hat{P}$ is a deployment strategy, each item (P_k, Q_k) in $\hat{P}$ must satisfy the condition $(P_k \cap Q_k) = \phi$ (see Subsection II-A). This can be extended to item $(P_k[i] \cap Q_k[i]) = \phi$ for all i and k by

$$(P_k \cap Q_k) = \phi$$

$$\Leftrightarrow [\bigcup_i (P_k \cap U^{(i)})] \cap [\bigcup_{i'} (Q_k \cap U^{(i')})] = \phi$$

$$\Leftrightarrow (P_k \cap U^{(i)}) \cap (Q_k \cap U^{(i')}) = \phi \quad (9)$$

From Equation 9, for each host s_k and each pair of SDG $G^{(i)}$ and $G^{(i')}$, we have

$$P_k[i] \cap Q_k[i'] = \phi$$

which implies that the matrix $\hat{D}$ satisfies the inter-application integration property.

Lemma 4.4: In the SOA environment with tree-structured network, if $\hat{P}$ is a minimum cost S-deployment for unified graph $\hat{H}$ and $\hat{D}$ is a minimum cost M-deployment for the SDGs ($H^{(0)}$,$H^{(1)}$, . . . ,$H^{(N)}$), then, $cost(\hat{P}) = cost(\hat{D})$.

Proof: According to Theorem (4.1), $\hat{P}$ is an S-deployment of $\hat{H}$ if and only if $\hat{D}$ is an M-deployments of ($H^{(0)}$,$H^{(1)}$, . . . ,$H^{(N)}$). Consider the cost of $\hat{P}$ which is equal to $cost(\hat{P}) = \sum_k cost((P_k, Q_k))$. Since the *net* (or *comp*) cost of node x in $\hat{H}$ is the sum of the *net* (or *comp*) costs of the node x in all R-graphs ($H^{(0)}$,$H^{(1)}$, . . . ,$H^{(N)}$) that comprise the graph $\hat{H}$ Thus, $cost((P_k, Q_k)) = \sum_{1 \le i \le N} cost((P_k[i], Q_k[i]))$. Finally, since $\hat{D}$ $=\{ P_k[i], Q_k[i] \mid 1 \le i \le N, 0 \le k \le M \}$, thus, $cost(\hat{P}) = cost(\hat{D})$ and the lemma follows.

Based on Theorem (4.1) and Lemma (4.4), Algorithm MINIMUM_M_DEPLOYMENT finds the minimum cost M-deployment in terms of application group and tree-structured network.

Time Complexity: The unified graph $\hat{H}$ has no more than $\hat{n} = \sum_i |U^{(i)}|$ nodes and $\hat{m} = \sum_i |F^{(i)}|$ edges. Thus, finding the minimum S-deployment for SDG $\hat{H}$ takes $O(\hat{n}\,\hat{m}\,log(\hat{n}^2/\hat{m})$ time as mentioned in Subsection II-C. This cost is also the total time for finding the minimum M-deployment since this cost dominates the costs of other steps, such as unifying and projecting the nodes of SDGs.

CHAPTER 5 CONCLUSION REMARK

This paper investigates the problem of deploying *N* distinct applications over a tree-structured network. In previous research results, every application instance is deployed individually. The main contribution of this paper is to enhance the deployments of service components invoked by multiple applications. The collection of services in different applications but utilizes the same type of resources are deployed in a bundle in order to reduce the administration cost. This paper develops an efficient algorithm for finding the optimal deployment strategy that satisfies both the intra-application and inter-application integration properties.

REFERENCE

[1] L.B. Chen and I.C. Wu. "Detection of summative global predicates," *IEICE Transactions on Information and Systems*, vol. E86-D, no. 5, pp. 976-980, May 2003.

[2] W. Chu J.Y. Hong L.P. Chen, I.C. Wu and M.Y. Ho. "Incremental digital content object delivering in distributed systems," *IEICE Transactions on Information and Systems*, vol. E93-D, no. 6, June 2010.

[3] L.R. Ford and D.R. Fulkerson. "Flows in Networks," *Princeton Univ. Press, Princeton, NJ*, 1962.

[4] L.R. Ford and D.R. Fulkerson. "Maximal flow through a network," *Can. J. Math*, vol. 8, no. 4, pp.399-404, 1956.

[5] A.V. Goldberg. "Recent Developments in Maximum Flow Problems," *Technical report 98-045, NEC Research Institute, Inc.*, 1998.

[6] A.V. Goldberg and R.E. Tarjan. "A new approach to the maximum-Flow problem," *Journal of the ACM s*, vol. 35, no. 4, pp. 921-940, Oct. 1998.

[7] K.Watanabe H.Tamura K.Nakano and M.Sengoku. "The p-collection problem in a flow network with lower bounds," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. E80-A, no. 4, pp. 651-657, Apr. 1997.

[8] R.K. Ahuja T.L. Magnanti and J.B. Orlin. "Network Flows: Theory, Algorithms, and Applications," *Prentice-Hall*, 1993.

[9] A.Ehuchi S.Fujishige and T.Takabatake. "A polynomial-time algorithm for the generalized independent-flow problem," *J Oper Res Soc Jpn*, vol. 47, pp. 1-17, 2004.

[10] C.E. Leiserson T.H. Cormen and R.L. Rivest. *Introduction to Algorithms.* The MIT press, 1989.

[11] K.Li H.S.Francis Y.L.Chin and W.Zhang. "Multimedia object placement for transparent data replication," *IEEE Trans. Parallel and Distributed Systems, s*, vol. 18, no. 2, pp. 212-224, 2007.

[12] OASIS Standards. *Universal Description Discovery and Integration.* http://www.oasisopen.org/specs/index.php, 2009.

[13] N. Roussopoulos, "Materialized Views and Data Warehouses," *ACM SIGMOD Record*, vol. 27, no. 1, pp. 21-26, Mar. 1998.

[14] W3C. *Web Services Description Language.* http://www.w3.org/TR/wsdl, 2008.

[15] Wikipedia. *Service-oriented architecture.* http://en.wikipedia.org/wiki/Serviceoriented_architecture, 2009.

[16] I.C. Wu and L.B. Chen. “On detection of bounded global predicates,” *The Computer Journal*, vol. 41, no. 4, May 1998.

[17] X.Tang and S.T.Chanson. “Minimal cost replication of dynamic web contents under flat update delivery,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 5, pp. 431-441 May 2004.