RC24437 (C0711-044) November 27, 2007
Computer Science

IBM Research Report

Automatic Model-Based Service Hosting Environment Migration

Liang Liu, Li Ying, Qian Ma, Ke Wei Sun, Ying Chen, Hao Wang
IBM Research Division
China Research Laboratory
Building 19, Zhouguancun Software Park
8 Dongbeiwang West Road, Haidian District
Beijing, 100094
P.R.China

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Automatic Model-Based Service Hosting Environment Migration

Liang Liu, Ying Li, Qian Ma, Ke Wei Sun, Ying Chen, Hao Wang
IBM China Research Laboratory, Beijing, 100094, P.R.C.
{liuliang, lying, maqian, sunkewei, yingch, wanghcrl}@cn.ibm.com

Abstract

Efficient and effective migration of system services is critical to cope with intrinsic-changed nature of Service-Oriented Architecture (SOA). However, due to the large amount of configuration items, complicated mapping and complex dependency relationship among system services, migrating into a new Service Hosting Environment satisfying the operation requirement of SOA becomes an error-prone and time-consuming task. The SCM project in IBM develops a novel approach to migrate Service Hosting Environment shaped in Unix-like systems. Firstly, this approach builds a set of configuration models for various system services. Then based on models, it presents knowledge based mapping to translate system service configurations between Service Hosting Environments. Finally, it designs a dependency hierarchy deducting algorithm to compute the dependency relationship among system services for migration traceability and error determination. A SCM prototype has performed well on largely reducing time, labor and errors in real migration cases.

1. Introduction

SOA is considered as a promising approach to construct large-scale enterprise information systems spanning the Internet [1]. For optimizing performance, improving flexibility and availability etc. to meet open and dynamic business requirements, a SOA system should be resilient enough to cope with the challenges which usually leads to migrate underlying system services [2] because a SOA system heavily relies on these services. For instance, the AIX Mail Service might be used by a web service that processes purchase orders to exchange messages between users. The authentication of WebSphere Application Server and DB2 database is also heavily dependent on the AIX User/Group Management Service.

However, migrating Service Hosting Environment is a time-consuming and labor-intensive work which requires high skills. During migration, their exist large amount of configuration items, complicated configuration item mapping and complex dependency relationship among system services which greatly increases the obstacles to detect the errors. So it is hard for system migration engineers (SME) to take everything into consideration. And according to IDT data, migration cost is up to 13% of IT management cost in the past years [3].

To address the above challenges, the SCM project is developing a novel approach to migrate Service Hosting Environment shaped in Unix-like systems. Figure 1 gives an overall architecture of SCM. In this architecture, Discovery Agent discovers and captures configuration items of system services according to CIM-based system service configuration models in source Service Hosting Environment. Migration Assistant provides discovery guides from SMEs to Discovery Agent, builds the dependency relationship tree among system services to determine the migration sequence for migration traceability and error detection, queries Migration Advisory Database to get knowledge-based mapping to finish mapping and build the appropriate migration plan/process. Provisioning Agent provisions mapped configurations of system services in target Service Hosting Environment to build same or similar entironment for the toppers.

The rest of the paper is organized as follows.

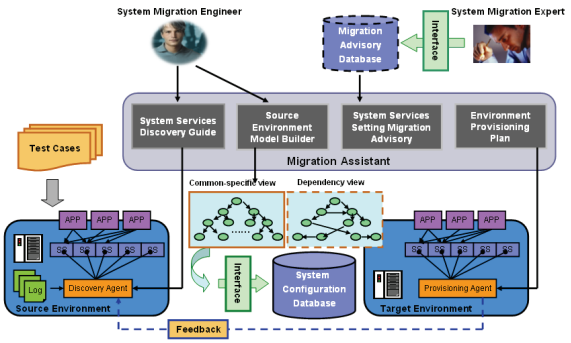

Figure 1 Overall System Architecture

Section 2 describes how to build configuration models of system services. Section 3 presents knowledge-based configuration item mapping. Section 4 discusses a system service dependency hierarchy deducting algorithm. Section 5 briefly discusses related work. Section 6 concludes the paper.

2. System Service Configuration Model

Service Hosting Environment could be regarded as being made up of a set of system services whose function and behavior are driven by their configurations respectively. And almost all system services in Unix-like systems are homologous with same or similar configuration syntax format and semantics. For example, most configuration items in DNS client configuration files (*/etc/inetd.conf*) in different Unix-like systems are same.

Using such features and taking configuration entry format (such as inetd.conf's configuration entry format: *ServiceName SocketType Protocol WaitStatus UserName ServerProgram ServerProgramArguments*) as an input, Source Environment Model Builder in SCM can automatically generate initial CIM-based hierarchy models for similar or same system services in different Unix-like systems according to the related configuration syntax format and semantics. The root model includes configuration items of a system service , each of which is shared in function and semantic by all Service Hosting Environments. Non-root models include configuration items of a system service shared by one or some Service Hosting Environments. From the root model to a special child model, a full configuration of a system service in a specific Service Hosting Environment can be abstracted. SMEs can use the tool to further enhance or improve the generated models to meet practical requirements with their expert knowledge. Figure 3 illustrates automatically generated SCM_InetdConfCommonModel and its child can effectively represent the configuration of inetd service in Tru64, Solaris, AIX, Redhat and HPUX.

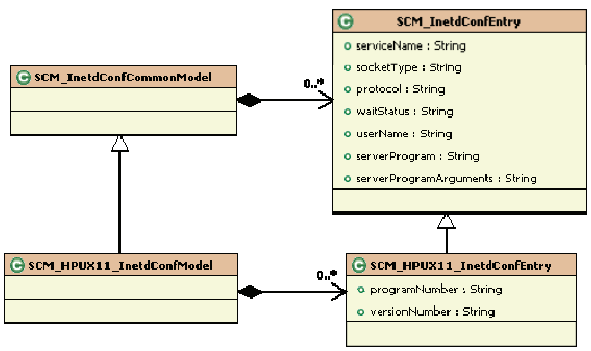

Figure 2. inetd Configuration Model

3. Knowledge-Based Mapping

Based on configuration models for system services, the possible configuration item mapping can be summarized as follows:

Let: M_A={ C_1, C_2, …, C_l, A_1, A_2, …, A_m } represents the configuration model of a system service S in a source environment.

M_B={ C_1, C_2, …, C_l, B_1, B_2, …, B_n } represents the configuration model of S in a target environment.

C_i represents a common configuration item.

A_j and B_k represent specific items in M_A and M_B.

$F(I_1, I_2, \ldots I_o)$ represents the combination of the configuration items (I_1, I_2, …I_o) can meet a specific purpose or function. Then:

- One to One Mapping

If $F(A_i)=F(B_j)$, A_i can be mapped to B_j directly.

- One to N Mapping

If $F(A_i)=F(B_1, B_2, B_3,\ldots B_j)$, A_i can be mapped to the combination of B_1, B_2, B_3,…B_j.

- N to One Mapping

If $F(A_1, A_2, \ldots A_i)=F(B_j)$, the combination of A_1, A_2, …A_i can be mapped to B_j.

- M to N Mapping

If $F(A_1, A_2, \ldots A_i)=F(B_1, B_2, \ldots B_j)$, the combination of A_1, A_2,…A_i can be mapped to the combination of B_1, B_2, …B_j.

Obviously, it is the key that how to locate the mapping function F to apply these mapping. In our implement, such functions are abstracted from a set of mapping knowledge tables accumulated and defined by SMEs. Figure 3 presents an example configuration item mapping table for DNS resolv.conf. For non 1-to-1 mapping, assembled mapping can be defined in the table for the left column related CIs. These tables can be edited or refined after being introduced into SCM as XML files. With the guide of these mapping knowledge tables, SCM can automatically translate configuration items in models among different service hosting environment.

OS Name	Tru64 v5.0	HPUX v11.0	Solaris v8	AIX v5.1	AIX v5.3	Redhat EL v4
Conf. File Name	**/etc/resolv.conf**	**/etc/resolv.conf**	**/etc/resolv.conf**	**/etc/resolv.conf**	**/etc/resolv.conf**	**/etc/resolv.conf**
Conf. Syntax	nameserver *IP* … domain *dValue* search *sList* sortlist *sValue* options *oList* …	nameserver *IP* … domain *dValue* search *sList* options *oList* …	nameserver *IP* … domain *dValue* search *sList* sortlist *sValue* options *oList* …	nameserver *IP* … domain *dValue* search *sList* options *oList* …	nameserver *IP* … domain *dValue* search *sList* options *oList* …	nameserver *IP* … domain *dValue* search *sList* sortlist *sValue* options *oList* …
Configuration Items Mapping in CIM-Based Model						
String[] nameserver	nameserver *IP*	nameserver *IP*	nameserver *IP*	nameserver *IP*	nameserver *IP*	nameserver *IP*
String domain	domain *dValue*	domain *dValue*	domain *dValue*	domain *dValue*	domain *dValue*	domain *dValue*
String search	search *sList*	search *sList*	search *sList*	search *sList*	search *sList*	search *sList*
String sortlist	sortlist *sValue*		sortlist *sValue*			sortlist *sValue*
options *oList* …						
String ndots	ndots:*n*	ndots:*n*	ndots:*n*	ndots:*n*	ndots:*n*	ndots:*n*
boolean debug			true | false	true | false	true | false	true | false
String retry_attempts			retry:*n*			attempts:*n*
String retrans_timeout			retrans:*n*			timeout:*n*
boolean rotate						true | false
boolean no-check-names						true | false
boolean inet6						true | false

Figure 3 DNS Client Configuration Mapping Table

4. Dependency Relationship Deducing

It is clear that tight dependency relationship exists among system services. Migration Service Hosting Environment based on dependency relationship will be very helpful to escape possible errors or conflicts, trace the process, and restore the target environment.

Definition 1: ***Base Functional Unit (BFU)***

BFU is a functional operation which serves others for a specific purpose, such as user account management function in NIS. It is an atomic function of a system service. A system service maybe has more than one BFU. Suppose A is a system service, then $A=\{ a_1, a_2, a_3,\ldots\ldots, a_n \}$, where a_i is a BFU.

Definition 2: ***Input*** and ***Output*** **of a System Service**

For A, a request for its service (RequestForServe) is defined as an Input and a request for others to serve it (RequestForServed) is defined as an Output, which can be represented as $F_{in}(A)$ and $F_{out}(A)$ respectively. Based on Definition 1, the following exists:

$F_{in}(A) \subseteq \cup F_{in}(a_i)$, $F_{out}(A) \subseteq \cup F_{out}(a_i)$.

Definition 3: ***Dependency Relationship R***

A ***R*** B represents that A has a request to be served by B, which can be expressed as below:

$A \boldsymbol{R} B = \{ (a_i,b_j) \mid \exists (f \in F_{out}(a_i) \wedge f \in F_{in}(b_j)), a_i \in A, b_j \in B \}$

If given A, B, C, while $\exists$((a,b) $\in$ A ***R*** B and (b,c) $\in$ B ***R*** C), it can be deducted that A has an indirect dependency relationship with C.

Definition 4: ***Dependency Loop L***

Suppose a_1, $a_s \in A_1$, $a_2 \in A_2$, …$a_j \in A_j$ …$a_n \in A_n$, A_j is a system service, if the following condition is satisfied:

$(a_1, a_2) \in (A_1 \boldsymbol{R} A_2)$,…, and $(a_{j-1}, a_j) \in (A_{j-1} \boldsymbol{R} A_j)$,…, and $(a_{n-1}, a_n) \in (A_{n-1} \boldsymbol{R} A_n)$ and $(a_n, a_s) \in (A_n \boldsymbol{R} A_1)$

Then a ***dependency loop L*** exists among A_1 to A_n. If such a Loop exists, all related system services can be treated as a system service in logical and each system service will be treat as a BFU respectively.

Dependency Hierarchy Deducting Algorithm

This algorithm is used to build a depth tree to represent the dependency hierarchy of system services, in which parent nodes directly depend on children nodes and the depth of a node represents its level in dependency relationship. Its pseudo-code is as follows:

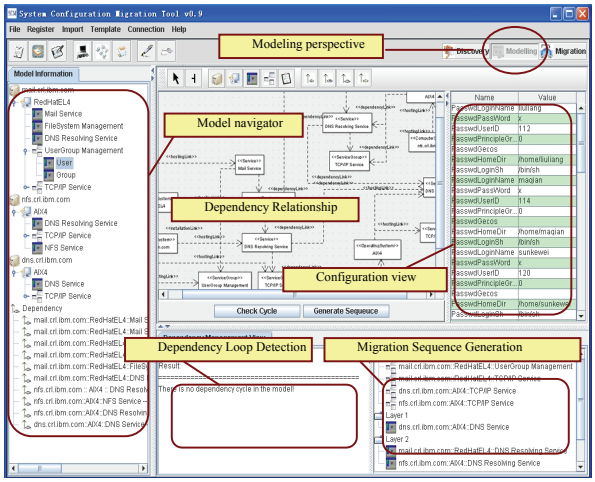

Figure 4 System Services Dependency Relationship UI

1. Create an initial Tree *T*: an application, App is selected as the root node with depth ←0 and its directly dependent system services A[1…n] with each depth←1;
2. i ←1
3. CURRENT_NO ← n
4. REPEAT
   a) j←1
   b) WHILE (j<= CURRENT_NO)
      i. A_j← a system service at ith depth
      ii. IF(A_j directly depends on system services B[1…m])
      THEN FOR t ← 1 to m
      IF (B[t] $\notin$ *T*){
      [Add B[t] to *T* as a child of A_j]
      [Set depth of B[t] ← i+1]
      }ELSE IF !(Check Dependency Loop from A_j to B[t]){
      [Add B[t] to *T* as a child of A_j]
      [Update depth of B[t] ← i+1]
      [Update depth of B[t]'s sub trees]
      }ELSE{
      [Construct a virtual system service, C including nodes in the loop]
      [Update depth of C ← depth of B[t]+1]
      [Update parent-child relationship of nodes having same depth with B[t]]
      [Update depth of sub trees of each node in C]
      [Record the loop and Give an alert]
      }
      iii. j++
   c) i++

d) CURRENT_NO←the number of system services at the ith depth

5. UNTIL CURRENT_NO = 0
6. RETURN *T*

5. Related Work

The CHAMPS [6] project proposes a "best of breed" approach by leveraging existing techniques, such as workflow etc, for an integrated Change Management solution of software deployment. Regarding the intrinsic-changed nature of SOA, an autonomic computing approach is proposed by Li et al. [7] to reconfigure web services in a distributed environment by dynamically provisioning and re-provisioning their instances to meet Service Level Agreement (SLA). While these work target the deployment of web services and software in the higher layer, our approach focuses on constructing underlying Service Hosting Environment for upper application.

Vmware P2V [8] is an industry tool targeting the similar scenario as this paper, which automatically migrate a real operating system into a Vmware virtual machine. It is different from ours that it merely targets the same operating system, while our tool involves the configuration mapping to support the migration among heterogeneous systems. In addition, the migration in the P2V tool actually is an image copy, while ours can support the selective migration of system services, and further use the dependency management mechanisms to guarantee the migration's validity.

An improved approach [9] to generate configuration files from a database is proposed by Jon Finke etc. This method can be used to keep the data extraction and the formats of the file in the database, while doing the actual file generation and version control management on target systems.

Automatically extracting dependencies among system components is considered in [10]. This technique can be used to complement our work by automatically capturing the dependencies between a specific application and the system services.

6. Conclusion

For quickly building resilient Service Hosting Environment to meet demand business requirements, this paper develops a novel approach to migrate system services shaped in Unix-like systems. It builds a set of configuration models for various system services, presents knowledge based mapping to migrate system service configurations and design a dependency hierarchy deducting algorithm to compute the dependency relationship among system services for migration traceability and error determination.

Several extensions to the present work are planned. We are trying to integrate change management with IBM Configuration Management Database (CMDB) into our approach following ITSM processes. Another challenging research direction is configuration optimization in target Service Hosting Environment.

7. References

[1] K.ishore Channabasavaiah, Kerrie Holley, Edward M. Tuggle, Migrating to a service-oriented architecture. IBM Corporation, White paper, 2003, http://www.ibm.com/developerworks/webservices/library/ws-migratesoa/

[2] Matthew Eastwood, Jessica Yang, Server Migration 2004: Understanding Platform and Workload Impacts, IDC Market Analysis, 2004.

[3] Dell 's OpenManage Approach to System Management: Simplification Through Standards and Partnerships: http://www.dell.com/downloads/global/corporate/iar/20050901_idc.pdf

[4] Mark Melenovsky, Jessica Yang, IBM Director: Driving Efficiencies in Scale-Out Computing, IDC White Paper, 2004.

[5] Heinz-Gerd Hegering, Sebastian Abeck, Bernhard Neumair, Integrated Management of Networked Systems: Concepts, Architecture, and Their Operational Application, Morgan Kaufmann Publishers, 2006.

[6] A. Keller, J.L. Hellerstein, J.L. Wolf, K.-L. Wu, V. Krishnan, The CHAMPS System: Change Management with Planning and Scheduling, In 9th IEEE/IFIP Network Operations and Management Symposium (NOMS'04), 2004.

[7] Ying Li, Kewei Sun, Jie Qiu, Ying Chen, Self-Reconfiguration of Service-Based Systems: A Case Study for Service Level Agreements and Resource Optimization, In 2005 IEEE International Conference on Web Services (ICWS'05), 2005.

[8] VMware P2V Assistant, available at http:// www.vmware.com/pdf/p2v_datasheet.pdf

[9] Jon Finke, An Improved Approach for Generating Configuration Files from a Database, Proceedings of the 14th USENIX conference on System administration, Pages: 29-38, 2000, USENIX Association Berkeley, CA, USA.

[10] A. Keller, G. Kar, Dynamic Dependencies in Application Service Management, In International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'00), 2000.