제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

분산 시스템 환경에서의 컨텐츠 및 어플리케이션의 디플로이먼트

윤태웅*, 안형근*, 최은미**, 민덕기*
*건국대학교 컴퓨터정보통신공학과
**한동대학교 전산전자공학부
*{taewoong, h96ahg, dkmin}@konkuk.ac.kr, **emchoi@handong.edu

Contents and Application Deployment in Distributed System Environment

Tae-Woong Yun*, Hyung-Geun An*, Eunmi Choi**, Dugki Min*
*Dept. of Computer Science and Engineering, Konkuk University
**Dept. of Computer and Electrical Engineering, Handong University

요 약

인터넷 사용자가 증가함으로서 웹서버와 같은 서버들의 부하를 분산하기 위한 분산 시스템환경이 사용되어지고 있다. 이러한 경우에 있어서 기존의 ftp와 같은 도구로는 웹서버들간의 컨텐츠(Contents) 또는 애플리케이션(Application)을 동기화 함에 있어 한계가 나타난다. 이 논문에서 소개하는 Deploy 시스템은 이러한 분산 시스템 환경 하에서 시스템간의 컨텐츠 또는 애플리케이션의 동기화를 관리하는 방법을 제시한다. 시스템에서는 배포될 노드들을 클러스터라는 그룹단위의 배포가 가능하며 배포할 컨텐츠 또는 애플리케이션을 패키지화하여 버전관리, 히스토리 관리(백업), 스케쥴링을 통한 예약 작업이 가능하다. 또한 현재 상용화 되어있는 분산 관리 시스템과의 연동으로 통해서 보다 효과적인 분산 관리 배포 시스템을 가능하게 한다.

1. 서론

현재 분산시스템을 사용하는 분야 중 가장 두드러진 분야는 웹서버 분야, 애플리케이션 서버 분야 일 것이다. 이러한 분산환경으로 컴퓨팅 환경이 변화되면서 발생한 문제들 중의 한가지는 시스템간의 컨텐츠(Contents) 또는 애플리케이션(Application)의 동기화(Synchronization) 문제이다.

본 논문에서는 분산 컴퓨팅 환경에서 시스템들의 컨텐츠 또는 어플리케이션의 동기화를 관리하기 위한 Deploy 시스템을 소개한다. 이 시스템에서는 동기화가 필요한 각각의 노드(시스템)들을 하나의 논리적 단위인 클러스터(Cluster)로 나누어 그 클러스터에 속한 모든 노드 혹은 일부의 컨텐츠, 애플리케이션을 배포가 가능하도록 하며 각 노드들에 설치된 컨텐츠나 어플리케이션의 버전을 통하여 각 서비스가 동일한 버전을 제공하도록 동기화를 제공하고 있다.

배포할 컨텐츠 및 애플리케이션은 XML Configuration을 사용하여 패키지(Package)화하여 버전(Version)관리 및 히스토리(History) 관리가 가능하다. 또한 스케쥴링(Scheduling) 기능을 제공하여 배포작업을 예약 실행 할 수 있다. 자바로 개발되어 시스템 및 운영체제에 독립적으로 운영될 수 있으며 분산환경에서 서버와 에이전트(Agent) 통신(TCP/IP 프로토콜, Java RMI)을 기반으로 동작한다.

본 시스템은 현재 가상의 클러스터링으로 동작하기도 하지만 현재 상용화해서 판매중인 분산 클러스터 시스템인 NexCenter[1]와도 연동되어 운영될 수 있는 구조를 가지고 있다.

2. Deploy System 기능

컨텐츠나 어플리케이션을 디플로이 서비스를 제공하는 본 시스템은 다음과 같은 기능을 제공한다.

- 시스템간의 배포를 일괄적으로 수행 할 수 있다.
- 배포할 컨텐츠 또는 애플리케이션을 하나의 단위(Package)로 관리
- 패키지 버전관리를 통한 컨텐츠 또는 애플리케이션의 히스토리(백업) 관리가 가능하다.
- 현재 노드에 설치된 컨텐츠 또는 애플리케이션 패

키지의 버전확인을 통하여 각 노드들간의 동기화를 확인 할 수 있다.

- 배포될 시스템을 클러스터단위로 관리 가능
- 스케줄링 기능을 통한 배포 작업을 예약 자동화
- 서버 시스템의 환경정보관리(configuration management)는 XML을 통해서 외부에서 손쉽게 수정이 하게 하고 시스템이 이를 바로 적용 할 수 있게 한다.
- GUI 형태의 관리콘솔(Management Console) 프로그램을 제공한다.

3. Deploy System 구조

Deploy System은 그림1 같이 구성된다. 클러스터를 이루는 한 개 이상의 노드(Real System)가 존재하며 패키지관리 및 버전 관리, 스케줄링을 위한 StagingServer가 있고 테스트 서버인 QAServer가 존재한다.

3.1 클라이언트 서버 구조적 관점

본 Deploy System을 클라이언트 서버 관점에서 보면 그림 1 와 같다. 서버측면에서는 위치적으로 3가

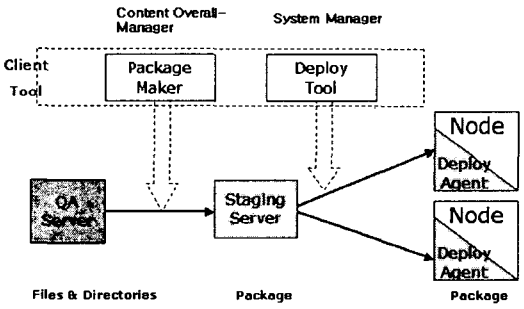

그림 1 Deploy System 클라이언트 서버 구조 관점

지의 단독 실행 가능한 서버가 있다. 그 3가지에는 QAServer, StagingServer, 그리고 노드에서의 Deploy작업을 대리하는 DeployAgent가 있다. 클라이언트는 크게 2가지 도구로 나누어진다. QAServer에서 파일이나 디렉토리를 옮기거나 패키지를 만들 수 있고 패키지의 버전을 관리하는 PackageMaker가 있고, StaginServer에 있는 패키지를 노드에 설치하거나 패키지 설치의 스케줄링 작업을 할 수 있는 DeployTool이 존재한다. 각각 PackageMaker는 전체 컨텐츠 관리자들이나 전체 어플리케이션 관리자들이 병렬적으로 접근 제어하며 DeployTool은 시스템 관리자 한 명이 일정한 시간에 설치하게 된다.

3.2 데이터의 이동 관점

그림2 에서는 데이터 이동으로 표현하고 있다. 먼저

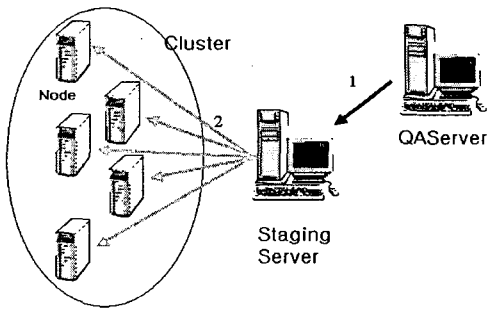

그림 2 Deploy System 데이터 이동흐름도

QAServer에 노드에 설치하려는 컨텐츠 및 어플리케이션이 존재하게 되며 여기서는 아직 패키지의 단위로 있지 않다. QAServer에 있는 데이터(Low Data)를 StagingServer에 옮기면서 패키지를 만들게 된다(번호(1) 참조). 기존의 패키지를 상속받을 수도 있으며, 새로운 패키지를 만들 수도 있다. QAServer에 있는 컨텐츠나 어플리케이션이 StagingServer에 패키지로 존재하게 되면서 여러 가지 관리가 가능하게 되는 것이다. 오 완성된 패키지는 StagingServer에서 실제 클러스터나 노드에 설치되는 것이다(번호(2) 참조). 즉, (1)의 상태는 패키지를 이루기 전인 파일이나 디렉토리로 이동하게 되나 (2)의 경우는 패키지라는 단위로 이동하게 된다. 패키지로 처리하면서 생기는 장점에 대해서는 Version Controller절에서 자세히 설명한다.

4. 시스템 컴포넌트

Deploy System을 구성하는 컴포넌트들에 대하여 자세히 살펴보자.

4.1 StagingServer

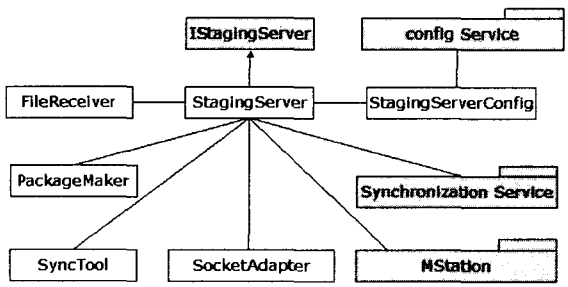

그림 3 StagingServer structure

메인 클래스는 StagingServer이며, IStagingServer를 통해서 RMI 인터페이스를 노출시키며,

FileReceiver를 통해서 QAServer로부터 파일을 전송 받는다. PackageMaker, SyncTool을 Facade Pattern을 사용해서 각각 PackageMaker Tool, Deploy Tool 클라이언트의 대응되는 일을 SocketAdapter를 통해서 TCP입력을 받아서 위임처리하며, StagingServerConfig는 StagingServer의 환경정보 클래스 이며 config service를 통해서 객체대 XML파일로 매핑된다.

Synchronization Service를 통해서 연결된 DeployAgent와의 동기화 처리를 한다. 또한 StagingServer는 NexCenter와의 연동시 MStaiton에 연결 가능한 구조는 가진다.

4.2 QA(Quality Assurance)Server

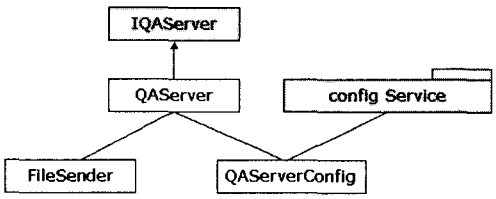

그림 4. QAServer structure

StagingServer와 마찬가지로 IQAServer가 RMI 인터페이스를 노출하며, FileSender는 StagingServer의 FileReceiver와 연결되어 클라이언트에서 PakcageMaker Tool을 사용해서 파일을 전송할 때 실제 파일 데이터를 전송하는 역할을 담당한다.

4.3 NodeAgent

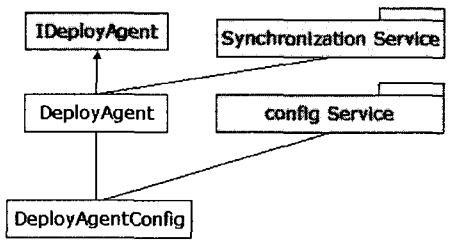

그림 5 DeployAgent structure

IDeployAgent는 DeployAgent의 RMI인터페이스를 노출하며 Synchronzation service를 사용해서 클라이언트에서 DeployTool을 사용해서 설치하려 하거나 스케쥴링을 해서 패키지를 설치 할 때 패키지를 전송하는 역할을 한다.

4.4 PackageMaker

PackageMaker의 주요 기능은 아래와 같다.

- 패키지 정보를 갖고 패키지를 만든다.
- QAServer에 있는 파일이나 디렉토리를 StagingServer의 패키지 내에 전송한다.
- 패키지 내에 파일을 삭제, 변경하며 디렉토리를 삭제, 변경, 생성한다.
- 파일혹은 디렉토리 전송을 모니터링 한다.
- 현재 패키지 버전을 백업하거나 기존의 패키지 상속해서 패키지를 만들거나 혹은 새로운 패키지를 만든다.

4.5 DeployTool

DeployTool의 주요 기능은 다음과 같다

- 하나의 패키지를 하나 이상의 노드나 클러스터에 설치 한다. 설치 시에는 기존의 서비스를 중간하고 하거나 기존의 가동중인 서비스에 관계없이 할 수 있는 옵션이 있다.
- 패키지의 설치 작업을 작업과 시간을 정해서 정해진 시간에 설치 할 수도 있고 정해진 기간만큼 정해진 반복 기간에 따라서 동기화를 할 수 있다.

5. 구현 이슈

Deploy System을 구성하는데 있어서의 기술적 이슈에는 Synchronization, Package structure, Version Control, Scheduling이 있다.

4.2 Synchronization

Synchronization Service는 단독으로 존재하는 서비스(Replication Engine)이며 한쌍의 Replication Engine 사이의 데이터를 동기화 한다.

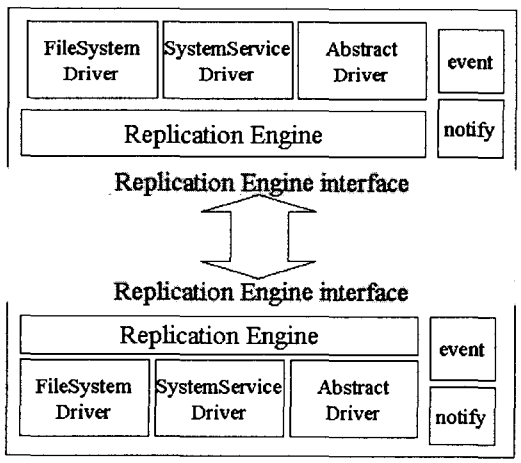

그림 6 Replication Engine structure

그림6에서와 같이 하나의 Replicaton Engine에는 다른 Replication Engine의 접근을 위한 인터페이스

를 가지고 있으며, 동기화를 위해서 필요로 하는 아이템을 복제(Replication)함으로써 같은 아이템을 동기화 하게된다. 인터페이스를 노출 하기 위해서 Replication Engine 들은 RMI서버 형태로 구현되어 있다. 따라서 Replication Engine 단독(단독 서비스)으로 상대편의 Replication Engine과 연결이 가능하다.

디폴트 드라이버로는 파일 시스템을 동기화하기 위한 FileSystemDriver가 있으며, 시스템 서비스 (내부 어플리케이션)의 동기화를 위한 SystemServiceDriver가 있다. 또한 확장 가능한 구조를 위해서 추상 드라이버를 제공해서 구현자가 여러 가지 드라이버를 확장 할 수 있는 구조를 가진다.

동기화 과정에서 감시하고 있는 아이템(파일이나 어플리케이션)이 변경되었을 때 그 이벤트를 Replication Engine에 전달하기 위한 event , 동기화 모드인 push와 pull 모드 지원을 위한 notify가 있다.

동기화의 과정에서 각 Replication engine, 즉 각 노드의 아이템의 비교의 과정에서는 은 IHaveList, UpdateList, ActionList 구조의 XML 표현 가능한 정보구조의 비교, 처리로 이루어진다. 또한 파일 시스템의 경우 파일의 수 가 많은 경우 비교 분석의 시간상의 효율성을 위해서 MD5 해쉬 함수가 사용되어진다.

5.2 Package

패키지는 Deploy를 하기 위한 논리적 단위의 묶음으로 컨텐츠의 경우는 특정 파일 혹은 디렉토리의 집합이 되며, 어플리케이션의 경우 어플리케이션의 집합이 된다. 패키지가 가져야 할 부가정보로는 패키지 생성일, 전체 사이즈, 아이템 개수, 버전 정보 등이 있다.

5.3. Version Controller

버전 관리에서는 현재 패키지를 정해진 백업 위치에 복사해 놓는 작업을 한다. 각 패키지는 공유의 버전을 가지며 이 버전은 변화가 가능하다. 따라서 하나의 패키지에 대한 버전 히스토리 관리가 필요하게 된다. DepolyTool에서는 특정 패키지의 특정 버전을 설치하거나 동기화 하는 것도 가능하게 된다.

5.4. Synchronization Scheduler

패키지를 노드 혹은 클러스터에 동기화 할 때 지정된 시간에 지정된 시간 간격만큼 동안 동기화를 자동화하는 일은 중요하다. 스케쥴링을 위해서는 정해진 절대시간 , 시간간격을 나타내는 태스크와 동기화 하려는 패키지, 노드 혹은 클러스터의 이름 , 동기화 옵션, 즉 액션을 입력해야 한다. 스케쥴링은 여러 개의 태스크를 지정된 시간에 실행해주는 일을 한다.

6. 결론 및 향후과제

Deploy System은 여러 개의 노드(시스템)에 컨텐츠나 어플리케이션을 복제(replication)하는 도구로써 클론드 서비스(Cloned Service)클러스터를 대상으로 한다. 그 구성은 3종류의 단독 서버와 2종류의 클라이언트로 구성되며, 노드들의 집합인 가상 클러스터 관리, 패키지 관리, 패키지 버전관리, 동기화 스케쥴러 관리를 하며, 각 서버의 환경정보는 문서의 표준 저장방법인 XML을 사용하였다.

향후에 추가할 부분들은 다음과 같다. Synchronization 의 Driver 확장으로 동기화 할 수 있는 아이템의 확장이 필요하다. 즉, 컨텐츠나 어플리케이션을 제외한 각 노드의 환경 변수나 초기화 변수, 그밖에 분산 관리 시스템에서 관리해야 하는 아이템을 Deploy System을 사용한다면 보다 효율적, 생산적인 관리가 가능하게 된다.

7. 참고문헌

[1] NexCenter available at URL : http://itdreams.co.kr/nex/index.htm

[2] Microsoft Application Center 2000 available at URL :http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/acs/proddocs/ac2k/acjmwe_welcome.asp

[3]Thanasis Loukopoulos, Ishfaq Ahmad and Dimitris Papadias "An overview of data replication on the Internet",Parallel Architectures, Algorithms and Networks, 2002. I-SPAN '02. Proceedings. International Symposium on , 2002 Page(s): 31 -36

[4] Tridgell, A., and MacKerras, P. "The rsync Algorithm", Technical Report, Department of Computer Science, Australian National University. 1996.

[5]Babaoglu, O., Bartoli, A. and Dini, G. "Replicated file management in large-scale distributed systems", Proceedings Workshop on Distributed Algorithms. 1-16. 1994

[6]Siegel, A., Birman, K., and Marzullo, K. Deceit "Deceit: a flexible distributed file system", Management of Replicated Data, 1990. Proceedings., Workshop on the , 1990 Page(s): 15 -17

[7] Chuck McManis ,"Take an in-depth look at the Java Reflection API " available at URL : http://www.javaworld.com/javaworld/jw-09-1997/jw-09-indepth.html