2000년 한국정보처리학회 추계 학술발표논문집 제 7 권 제 2 호

# 대규모 워크플로우 시스템을 위한 분산형 운용관리 도구의 설계 및 구현

이봉석 강태규 김광훈 백수기
경기대학교 전자계산학과
e-mail : {bslee, tgkang, kwang, skpaik}@kuic.kyonggi.ac.kr

# A Distributed Administration System For Very Large Scale Workflow Management System

Bong-Seok Lee, Tae-Gyu Kang, Kwang-Hoon Kim, Su-Ki Park
Dept. of Computer Science, Kyonggi University

요 약

최근에는 기업에서 처리되는 업무 형태가 그 절차도 더욱 많아지고 절차 간 흐름도 복잡해지고 있다. 여러 기업들이 관여되는, 이른바 대규모화 되어 가고 있다. 워크플로우 시스템은 업무를 이루고 있는 단위 업무 처리에 적합하도록 객체 단위로 설계되어 있다. 적용 분야와 경우에 따라서는 이러한 생성되어 수행중인 객체들이 수만에서 수백만개가 여러 기업에 분산되어 존재하기도 한다. 따라서 기존의 단일서버-클라이언트 형태의 관리 도구로는 이러한 대규모 시스템을 지원하기에는 서버 구조가 매우 비효율적이고, 서비스에도 많은 문제점을 안고 있다. 본 논문에서는 기업의 관리 모듈을 규모에 따라 워크플로우 엔진에 의존적으로 분산된 형태로 데이터와 기능들을 분산 배치시키고 메인 관리기에서 통합하여 관리함으로써 운용 서버의 부하를 줄이고, 가용성을 높일 수 있는 방안들을 추출하여 설계하고 구현하였다

## 1. 서론

워크플로우 기술이란 전자적 작업환경을 구현하기 위한 종합적인 연구 분야로서 궁극적으로 기존의 업무를 컴퓨터 및 네트워크를 통하여 자동화하려는 시도이다. 즉, 워크플로우 관리 시스템을 중심으로 한 조직내의 모든 업무 처리 절차의 통합을 구현함으로써 워크플로우 또는 업무 처리 절차가 주요 인프라 구조를 구성하며, 이는 결국 모든 업무의 자료화를 가능하게 할 뿐만 아니라 업무 처리의 이력 관리와 추적 조회 등의 운용관리 및 모니터링 기능을 효과적으로 지원하게 된다. 또한 조직의 성장에 따라 수반되는 정보 및 업무 처리의 복잡성 증가에 효과적으로 대응할 수 있어 조직의 확장성을 용이하게 한다. 이를 위해 운용적인 측면에서는 업무 처리를 담당하고 있는 수행 객체와 시스템 및 자원이 효과적으로 관리가 되어야 한다. 또한 시스템의 처리가 많은 부분을 차지하기 때문에 객체간에 통신은 복잡한 구조를 형성하며 이루어지게 된다. 워크플로우 엔진과 함께 주요 요소로 자리잡은 운용 도구는 대부분이 엔진의 수행 객체와의 통신을 통해서 제어 및 시스템 전반적인 관리를 한다. 업무가 복잡하고 다양하며 일 처리가 빈번한 업무의 형태를 지닌 조직일수록 운용자의 개입이 절실하며 이는 엔진의 수많은 수행 객체들과의 상호작용이 늘어남을 의미한다. 그러나 기존의 단일서버-멀티 클라이언트 형태로는 수많은 수행객체를 관리하거나 여러 운용자의 요구를 동시에 처리하기에는 매우 불안전하고 비효율적인 구조로 되어 있다. 본 논문에서 다루고 있는 운용 도구는 서버측이 분산 배치된 형태이며 이러한 문제들을 해결하고 있다. 2 장에서는 대규모 워크플로우 시스템에 대해서, 3 장에서는 분산형 운용도구를 기술하고, 4 장에서는 설계 및 구현에 대해서, 5 장에서는 결론과 향후과제를 설명한다.

## 2. 대규모 워크플로우 시스템

기업의 업무는 전자상거래의 처리절차와 같이 하나의 기업 프로세스가 여러 기업에 걸쳐 이루어지는 경

우가 많아지고 있다. 따라서, 하나의 프로세스를 이루는 단위업무(Activity)의 수도 많아지고 처리도 복잡하게 된다. 이는 그림 1 에 나타나 있다.

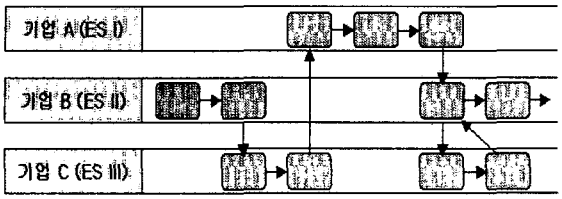


그림 1. 여러 기업에서 수행되는 프로세스

기업 B 에서 두 번째 단위업무까지 처리되고 기업 C 에서 처리되다 다시 기업 A 에서 일이 처리되는 간단한 예이다. 이러한 단위 업무를 스케쥴링하고 수행을 관리하는 워크플로우 엔진(Enactment Service)이 기업 간에 존재하며 그들 간의 정해진 절차에 의해 실행을 하게 된다. ES(Enactment Service) I~III 이 기업 A~C 에서 사용되고 있는 예이다.

## 3. 분산형 운용 관리 도구

운용 관리 도구는 정의된 프로세스나 실시간에 진행중인 프로세스, 또는 수행이 완료된 프로세스의 상황을 모니터링을 통해 추출된 정보를 바탕으로 사용자 또는 역할을 관리하거나 자원 관리, 프로세스의 진행에 관계하여 감독 명령하는 기능을 운용자에게 제공하는 도구를 의미한다.

### 3.1 확장 기능

프로세스 관리, 사용자와 역할 관리와 같은 기본 외에 자원 제어 기능을 분산환경을 고려하여 시스템 제어 기능으로 확장했으며 이것의 주요 기능으로는 하나의 시스템이 여러 개의 서버들로 구성이 된다면 각각의 서버에 위치한 수행 객체들의 수행상태에 따라 서버의 기능을 정지, 시작 시킬 수 있으며, 또한 로드 벨런싱을 고려해서 특정서버에 부화에 따라 생성객체의 최대수를 할당할 수 있는 기능을 추가하였으며, 기능 수행에 있어서 객체 중심으로 다시 기능을 분류하고 확장하였다. 설계 및 구현을 통해 작업 가능한 기능들은 프로세스 리엔지니어링 즉 비즈니스 프로세스 환경에 변동이 있으면 변동에 따라 반영할 수 있다. 시스템 관리를 구동기 모듈을 이용하여 서버 맵 구조의 환경 설정 파일을 생성하여 쉽게 설정을 조작할 수 있다.

### 3.2 분산 구조

운용관리 도구에서는 그림 2 와 같이 기업 내 워크플로우 엔진 관리기가 설치된 곳에 의존적으로 위치한다. 운용 서버들의 분산된 형태에 따른 관리를 위해 운용서버 관리기(Mgr)가 필요하며, 이는 분산된 데이터의 일치성과 운용자에겐 가용성을 제공하기 위한 메커니즘을 제공한다. 이는 엔진의 수가 많은 수로 증가되어도 같은 메커니즘으로 사용되어 질 수 있다.

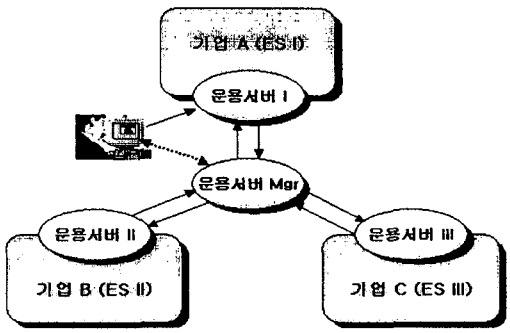


그림 2. 운용 서버의 분산 구조

### 3.3 데이터 분산

워크플로우 엔진에서 관리기(Workflow Process Manager)는 정의된 기업 업무로부터 인스턴스를 생성하게 된다. 하나의 인스턴스는 여러 개의 단위업무 인스턴스들로 구성된다. 이러한 정보들은 관리기에서 유지되는 정보에 포함되어 있다. 분산된 운용서버에서는 자신이 연결되어 있는 관리기로부터 이러한 정보들을 획득하여 관리한다. 운용 서버 Mgr 에서는 모든 기업에서 수행되고 있는 객체 정보가 유지되고 있다. 각각의 유지되는 정보는 그림 1 과 같다.

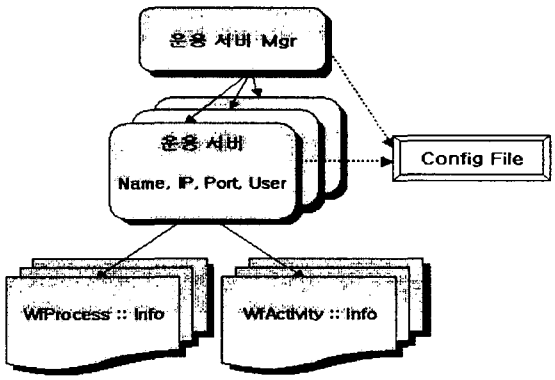


그림 3 분산된 데이터 구조

운용 서버 Mgr 에서는 전체 워크플로우 시스템에서 사용되는 JDBC Store 정보뿐만 아니라 워크플로우 서버가 구동 시 참조되는 Config 파일을 변경할 수 있으며, 구동 된 것들에 대해서는 각각 유지되는 정보를 관리한다. 즉, 운용 서버들과 워크플로우 엔진들에 대한 연결과 부팅을 할 수 있다. 아래 인스턴스들의 정보는 워크플로우 엔진으로부터 획득된다. 객체들에 대한 업데이트는 폴링 방식과 사용자 요청 시 정보갱신의 병행적인 방법을 사용함으로써 수동/자동 모두 가능하다. 운용 서버는 자신의 정보가 변경되면 상위 관리기에게 보고를 하여 같은 정보를 유지한다. 객체 레퍼런스 관리기는 엔진의 관리기를 통하지 않고 운용

적인 외부 제어가 필요한 객체를 직접 연결하여 요구를 하기 위함이다. 이는 엔진의 부하를 줄여줄 수 있을 뿐 아니라 그러한 정보를 바탕으로 다른 부가적인 기능 추가를 가능하게 한다.

### 3.4 대안적 분산 운용 서버

운용 관리 도구에는 서버 부분과 실제 운용자와의 인터페이스 역할을 하는 GUI 기반의 클라이언트 부분이 있다. 서버부분에서는 클라이언트로부터의 요청을 접수하고 수행한다. 엔진부분의 여러 컴포넌트와 통신을 하며 수행에 관여를 하는 핵심 역할을 담당한다.

#### 3.4.1 운용 서버의 구성

운용 서버는 운용자의 요구를 워크플로우의 엔진에 전달하고 결과를 통보해 준다. 엔진의 객체 레퍼런스 정보가 관리, 유지되며 데이터 베이스의 이벤트 히스토리를 관리한다.

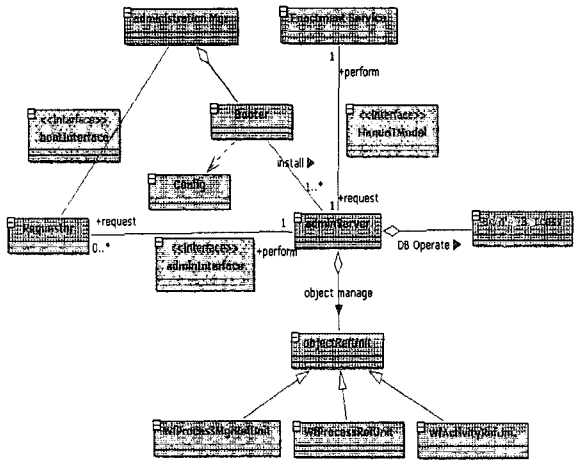


그림 4. 운용 관리 서버의 클래스 다이어그램

- adminServer : 클라이언트 요청과 수행을 담당
- objectRefUnit 수행 객체 레퍼런스 유지, 관리 담당
- adminDBaccess : 데이터베이스로의 접근/연산을 담당
- Booter : 운용 서버를 Config 파일에 따라 부팅

#### 3.4.2 클라이언트와 운용 서버와의 재연결

운용자(Requester)가 작업 중에 예기치 못한 이유로 서버가 다운되거나 연결을 잃게 되어 서비스가 불가능한 상황이 되었을 때 다른 경로로 재연결을 해주고, 계속적인 서비스를 제공 받을 수 있는 즉, 가용성을 향상 시킬 수 있는 메커니즘을 제공한다.

운용 서버 Mgr 은 Config 파일과 Booter 객체를 통해 운용 서버들, 엔진의 컴포넌트와 자동/수동으로 연결될 수 있고, 전체/부분으로 Start/Shut Down 할 수 있다.

운용 서버 Mgr 에서는 일치성 있게 서버들에 대한 정보와 그들이 관리하고 있던 정보를 유지하고 있다. 따라서 현재 접속중인 운용 서버에 문제가 발생되면 다른 운용 서버로 재연결이 가능하며, 문제가 생긴 운용서버는 문제 해결 후 재부팅을 한다.

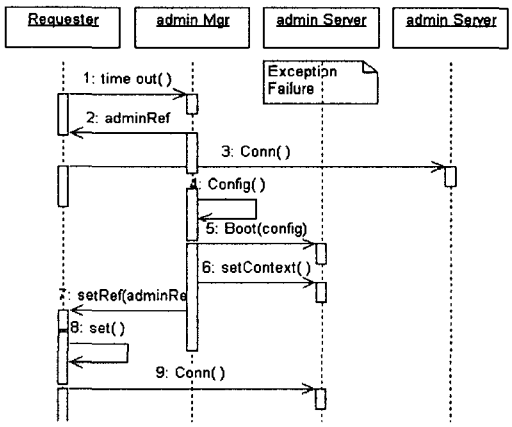


그림 5. 운용서버 문제 발생 시 재연결 과정

## 4. 구현 결과

분산 객체 관리의 표준인 CORBA 를 객체 간 통신 메커니즘으로 사용한다. 객체 호출은 CORBA 의 이름 서비스를 이용한다. 클라이언트와 운용서버 간, 운용서버와 엔진간의 통신을 위해 정의된 IDL 에 따라서 연동을 한다.

### 4.1 구현 환경

- WfMS : Hanuri/TFlow 2.0
- OS : Windows 95/98/NT, Solaris sparc 2.6
- ORB : javaIDL
- 구현언어 : JDK 1.2 이상
- 개발툴 : Jbuilder 3.0
- 데이터베이스 : Oracle 8.0.0.4.0.0

### 4.2 GUI 기반의 클라이언트

운용 도구에서는 기존의 워크플로우 제품들과는 다르게 운용자 전용 GUI 를 제작하였다. 워크플로우 시스템 구조 자체가 커지고, 다양하고 변화적인 업무들을 대단위 처리하는 트랜잭션 워크플로우 시스템에서는 운용자가 개입되어야 하는 경우가 늘고 있다. 기존의 방법은 런타임 클라이언트에 운용 기능이 내제되어 있는 형태로 많이 되어 있었는데, 이러한 기능들을 통합시킨 운용자 전용 클라이언트를 개발하였다. 사용적인 면에서 편의성을 제공하며 분리된 별도의 모니터링 서비스에 연결되므로, 확장성에서도 용이하다.

#### 4.2.1 객체 관리기 중심의 모니터링 서비스의 예

메인 프레임의 왼쪽에 트리 구조로 제공되는 것은 정의된 업무, 업무 인스턴스, 시스템을 이루는 컴포넌트들이다. 이것의 ID 를 모니터링 서버에 전달하면 요구한 정보를 얻어올 수 있다.

모니터링 서비스에 의한 정보들이 운용자 전용 GUI 에 제공되는 예제이다. 운용자는 이러한 정보들을 바

탕으로 워크플로우 시스템 내의 전반적인 요소들을 제어, 관리할 수 있다.

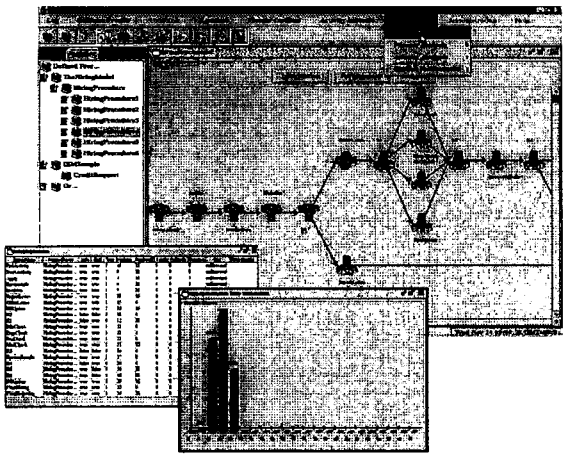

그림 6. 객체 관리기와 모니터링 서비스와의 연동 예

#### 4.2.2 운용 구동기의 예

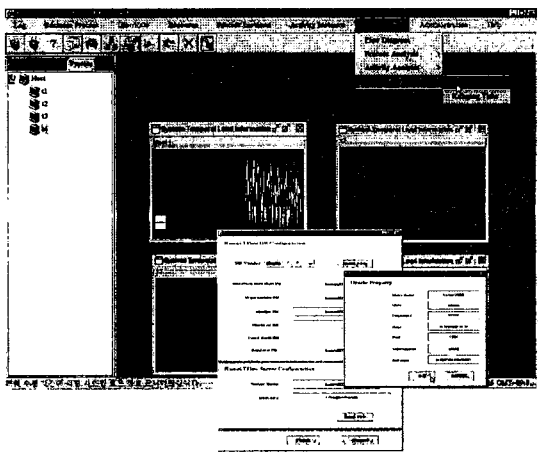

그림 7. 운용 구동기의 예

운용 구동기는 Config 파일에 대한 조작을 할 수 있고, Hanuri/TFlow 서버와 운용서 버들에 대한 Connection, Disconnection, Start, Shut down 연산을 할 수 있는 기능들을 제공하고 있다.

### 5. 결론 및 향후과제

기존의 단일서버-클라이언트 형태의 워크플로우 관리 도구로는 많은 기업이 연계되는 대규모 시스템을 지원하기에는 서버 구조가 매우 비효율적이고, 서비스에도 많은 문제점을 안고 있다. 본 논문에서는 기업의 엔진에 관리 모듈을 워크플로우 엔진에 의존적으로 분산되는 형태로 데이터와 운용 서버들을 분산 배치시키고 메인 관리기에서 통합하여 관리함으로써 부하를 줄이고, 가용성을 높일 수 있는 방안들을 추출하여 설계하고 구현하였다. GUI 기반의 전용 클라이언트를 운용자에게 제공함으로써 워크플로우 시스템 관리에 편의성을 제공한다. 향후, 웹에서 동작하여 운용자가 인터넷 환경에서 사용할 수 있도록 하고, Hanuri/TFlow 엔진에 의존적이지 않은 별도의 도구로써 사용되기 위한 작업들이 추가로 필요하다. 대규모보다 더 큰 규모인 초대형(Ultra Large Scale) 워크플로우 시스템에서 효과적인 동작을 하기위해서는 단순한 분산 구조보다는 더욱 정밀하게 멀티 레이어형태로의 형태의 구조가 바람직하다.

### [참고 문헌]


[1] http://www.wfmc.org/standard

[2] http://www.omg.org

[3] A. Elmagarmid, editor. ' Transaction Models for Advanced Database Applications' . Morgan-Kaufmann, 1992.

[4] A. Sheth and M. Rusinkiewicz, ' On Transactional Workflows' , in Data Engineering Bulletin ,16(2), June,1993.

[5] A. Silbercharz, H. F. Korth and S. Sudarshan, ' Database System Concepts' , McGraw Hill, 1997.

[6] A. Zhang, M. Nodine, B. Bhargava and O. Bukhes, ' Ensuring Relaxed Atomicity for Flexible Transaction in Multidatabase Systems' , ACM SIGMOD RECORD, Vol. 23, No. 2, 1994

[7] C. Mohan, D. Agrawal, G. Alonso, A. El Abbadi, R. Günthör, and M.Kamath. ' Exotica: A Project on Advanced Transaction Management and Workflow System' . ACM SIGOIS Bulletin, 16(1), 1995.

[8] Clarence A. Ellis and Gary J. Nutt, ' Modeling and Enactment of Workflow Systems' , in Proceedings of the 1993, June 1993.

[9] D. Geogakopoulos and M. Hornick, ' An Overview of Workflow Management : From Process Modeling to Workflow Automation Infrastructure' , International Journal of Parallel and Distributed Databases, Vol. 3, No. 2, pp. 119-153, 1995

[10] Kwang-Hoon Kim and Eun-Tae Won and Su-Ki Paik, 'A Hierarchical Framework for the Software Architecture Design of Workflow Management Systems', 경기대학교, 논문집(제 41 집 2 호) 별쇄본, 1997. 12.

[11] Dong-Soo Han and Hyang-Jae Park, " Design and Implementation of Web Based Business Process Automating HiFlow System" , Journal of KISS(C): Computing Practices, Vol. 4, No. 1, Feb. 1998