2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

워크플로우 시스템의 프로세스 구조 대안 비교

한상근*, 이도헌**
*전남대학교 정보통신협동과정
**전남대학교 전산학과
e-mail : skhan@dblab.chonnam.ac.kr

Comparison of Process Architectures for Workflow Systems

Sang-Keun Han*, Doheon Lee**
*Dept. of Information and Telecommunication, Chonnam National University
**Dept. of Computer Science, Chonnam National University

요 약

워크플로우 시스템은 다수의 클라이언트 및 서버가 연동하는 분산 시스템으로서 주어진 상황에 따라 구성 요소를 다양한 방식으로 분산시킬 수 있다. 본 논문에서는 워크플로우 제어를 클라이언트가 주도하는 형태와 서버가 주도하는 형태로 양분하여 장단점을 비교한다. 또한 클라이언트 주도형을 워크플로우 엔진을 배치하는 방식에 따라, 다시 중앙 집중형과 분산 처리형으로 구분한다. 이러한 프로세스 구조의 대안 분류 및 비교는 주어진 상황에서 최적의 워크플로우 프로세스 구조를 설계하는 데 필요한 지침을 제공한다.

1. 서론

워크플로우 시스템의 표준을 제정하는 기구인 WfMC(Workflow Management Coalistion)에서는 워크플로우를 '문서, 정보 또는 태스크가 한 구성원에서 다른 구성원에게 전해질 때 전적으로 혹은 부분적으로 특정한 규칙에 의해 비즈니스 프로세스의 자동화가 이루어지는 것'이라고 정의하고 있다. 이 정의에서 보듯이 워크플로우 시스템에서는 기업의 업무를 특정 규칙에 따라 그 흐름을 제어하고 관리하는 것이 핵심이라 할 수 있는데 그 역할을 담당하고 있는 컴포넌트가 바로 워크플로우 엔진이다. 워크플로우 엔진은 런타임 실행 환경을 제공하고 워크플로우 프로세스 인스턴스의 처리와 수행을 한다. 워크플로우를 수행하기 위해서는 실제로 워크플로우 엔진과 상호 작용을 통해 워크플로우 업무들을 처리하는 클라이언트 애플리케이션이 필요하다. 워크플로우 시스템을 구성하는 주요 컴포넌트는 이 외에도 프로세스 정의 도구, 관리 및 모니터링 도구 등이 있으나 본 논문에서는 워크플로우 엔진과 클라이언트 애플리케이션에 중점을 두었다.

본 논문에서는 한 클라이언트의 업무를 처리하기 위해 하나의 워크플로우 엔진 인스턴스를 생성하는 클라이언트 주도형 엔진과 하나의 프로세스를 처리하기 위해 하나의 워크플로우 서버 인스턴스를 생성하는 서버 주도형 엔진을 설계하고 비교한다. 본 논문의 구성은 다음과 같다. 우선, 2 장에서 워크플로우 시스템의 구성 요소와 워크플로우 엔진의 동작과정을 알아보고 3 장에서는 클라이언트 주도형 워크플로우 엔진과 서버 주도형 워크플로우 엔진을 비교하고 제안한다. 끝으로 4 장에서는 결론과 앞으로의 연구 방향을 제시한다.

2. 워크플로우 시스템의 구성요소와 엔진의 동작과정

2.1 워크플로우 엔진

* 본 논문은 한국과학재단(KOSEF) 특정기초연구(98-0102-11-01-3)에 의해 지원되었음.

워크플로우 엔진은 정의된 워크플로우를 수행하기 위해 다음과 같은 6 개의 컴포넌트를 갖는다.

- 클라이언트 모니터(Client Monitor)
  워크플로우 클라이언트가 엔진에 접근하기 위해 로그인을 하는지 감시한다. 클라이언트가 정상적인 로그인 과정을 마쳤을 경우에 로그인하는 클라이언트의 업무처리를 전담하게 될 워크플로우 엔진 인스턴스를 하나 생성한다.

- 프로세스 관리자(Process Manager)
  워크플로우 클라이언트로부터 워크플로우 수행 요청을 받고 워크플로우 업무 수행을 시작한다. 현재 수행중인 액티비티 수행이 끝났을 때, 프로세스 관리자는 다음에 수행될 액티비티를 스케줄링하고 그 액티비티를 액티비티 관리자에게 넘겨준다.

- 액티비티 관리자(Activity Manager)
  프로세스 관리자로부터 넘겨받은 액티비티의 수행에 필요한 참여자와 데이터가 수행 가능한 상태인지 확인하고 나서 클라이언트 측의 워크플로우 핸들러에게 액티비티를 넘겨준다. 실제적인 액티비티 처리를 담당한다.

- 참여자 관리자(Participant Manager)
  워크플로우 프로세스 정의기에 의해 참여자를 정의할 때 참여자는 임의의 조직이나 임의의 역할을 담당하는 집단, 특정 개인이 될 수 있다. 참여자 관리자는 그에 해당하는 적당한 수행자를 찾거나 정의된 참여자가 워크플로우를 수행할 수 없는 상황일 때에는 적당한 대리인을 선정하여 액티비티를 수행자를 찾아내게 된다. 또 클라이언트가 갖는 업무의 양에 따라 적당한 참여자를 찾아내기도 함으로써 참여자의 로드 밸런싱(load balancing)을 담당하기도 한다.

- 데이터 관리자(Data Manager)
  액티비티 수행에 필요한 문서나 데이터가 필요할 경우 그 데이터의 유효 여부를 확인한 후 데이터베이스에 저장된 데이터를 워크플로우 클라이언트에게 넘겨준다. 데이터베이스 접근자의 역할도 함께하여 워크플로우 정보를 얻어낸다.

- 이벤트/요청 처리기(Event/Request Handler)
  워크플로우 클라이언트 측의 워크리스트 핸들러와 통신하면서 어떤 이벤트가 발생했을 경우에 그 이벤트를 처리하기 위해 그 이벤트를 처리할 관리자에게 이벤트 처리를 요청하거나 때에 따라 클라이언트에게 이벤트 처리를 요청한다. 또, 클라이언트 측에서 작업 요청이 발생했을 경우에 그 요청을 수행할 컴포넌트를 찾아 그 요청을 넘겨준다.

2.2 워크플로우 클라이언트

각 클라이언트는 로그인을 통해 인증 과정을 거친 후에 엔진에 접근하게 되는데 다음의 2 개의 컴포넌트를 갖는다.

- 워크리스트 핸들러(Worklist Handler)
  로그인 한 수행자가 수행해야 할 액티비티들을 관리하는 역할과 클라이언트에서 실질적으로 워크플로우 엔진과 대화하는 역할을 한다. 엔진으로부터 할당받은 작업들과 관련 데이터들은 로그아웃을 할 때 객체 형태로 수행자의 로컬 시스템에 파일로 저장한다. 이렇게 하면 로그인을 할 때마다 자신의 액티비티가 기록된 데이터베이스를 모두 접근하지 않아도 되고 그에 따라 엔진의 부하를 줄일 수 있다.

- 워크플로우 클라이언트(Workflow Client)
  사용자와 워크플로우 엔진간의 대화 교류 매체로서 사용자는 클라이언트를 통해 업무를 할당받고 응용 프로그램을 호출, 업무를 처리한다.

2.3 워크플로우 데이터베이스

워크플로우 데이터베이스는 프로세스 정의기에 의해 정의된 워크플로우 프로세스 정의 정보가 저장되어 있다. 그 내부에는 프로세스의 구조, 프로세스를 구성하는 액티비티, 액티비티들의 흐름, 각 액티비티의 수행자, 액티비티의 흐름을 결정하는 조건, 작업에 필요한 데이터와 프로그램등의 정보를 갖는다. 또한 각 워크플로우 수행자들이 갖는 업무 리스트도 데이터베이스에 저장된다. 필요한 경우 업무 처리에 사용되는 문서 양식과 같은 데이터들은 별도의 데이터베이스에 저장해 두고 사용하기도 한다.

2.4 워크플로우 엔진의 동작 과정

워크플로우 엔진은 [그림 1]과 같은 동작의 흐름을 갖는다. 과정 ①에서 이벤트/요청 핸들러는 클라이언트의 워크리스트 핸들러로부터 워크플로우 작업을 요청받고 과정 ②에서 클라이언트로부터 받은 워크플로우 업무 요청을 프로세스 관리자에게 넘겨주어 워크플로우 업무 처리를 시작하도록 한다. 과정 ③에서 프로세스 관리자는 워크플로우 업무 수행에 필요한 액티비티를 선택한 후에 그 액티비티 수행 제어권을 액티비티 관리자에게 넘겨준다. 과정 ④에서는 액티비티 수행에 필요한 수행자를 참여자 관리자를 통해 얻어낸 후 과정 ⑤를 통해 적합한 수행자를 찾았는지의 여부를 알려준다. 유효한 수행자가 존재한다는 사실을 넘겨받은 액티비티 관리자는 액티비티 수행에 필요한 데이터가 무엇이 있는지 확인한다. 과정 ⑥을 통해 그 데이터의 유효성을 확인하고 데이터 관리자는 과정 ⑦을 통해 데이터의 유효성 여부를 반환하게 된다. 이 과정까지 성공적으로 마치게 되면 액티비티 관리자는 과정 ⑧을 통해 이벤트/요청 처리기에게 액티비티를 수행할 준비가 되어 있음을 알리고 과정 ⑨에서 이벤트/요청 처리기는 과정 ④를 통해 얻어진 수행자에게

처리해야할 새로운 액티비티가 할당되었음을 알린다.

위의 과정 ④와 ⑥에서 각각의 수행자와 데이터가 유효하지 않을 경우 워크플로우 업무 수행은 실패한다.

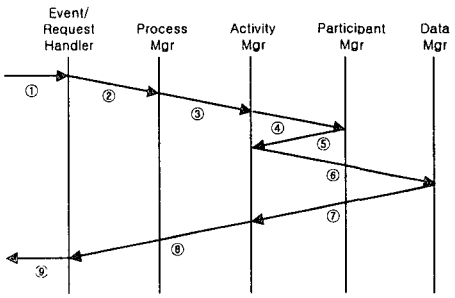

그림 1 엔진의 동작과정

3. 워크플로우 시스템의 프로세스 구조

본 논문에서는 워크플로우 클라이언트 애플리케이션과 클라이언트 시스템을 구분하기 위해 전자를 클라이언트, 후자를 클라이언트 머신이라고 정의한다.

3.1 클라이언트 주도형 엔진

이 장에서는 클라이언트 별로 하나의 워크플로우 엔진 인스턴스가 발생하는 두 가지 형태의 엔진 즉, 하나의 서버에서 쓰레드를 이용해 워크플로우 업무를 처리하는 중앙 집중형 워크플로우 엔진 형태와 뛰어난 클라이언트 머신의 처리 능력을 이용하여 과중한 워크플로우 엔진의 업무를 개인 시스템에 분담시키는 분산 처리형 워크플로우 엔진 형태를 비교한다.

3.1.1 중앙 집중형 구조

기존 워크플로우 엔진은 각 기능별 업무(예를 들어 스케줄링, 액티비티 준비, 데이터 관리, 참여자 관리 등)에 따라 서버에 독립적으로 컴포넌트로 존재하면서 유기적으로 동작하는 형태를 띄고 있었다[2]. 그러나 제안하는 중앙 집중형 워크플로우 엔진은 각 컴포넌트들이 하나의 프로세스로 묶여있는 형태를 갖는다.

3.1.1.1 중앙 집중형 워크플로우 엔진의 생성

클라이언트 모니터는 서버에 데몬(Deamon)형태로 클라이언트의 로그인을 대기한다. 클라이언트가 로그인을 통해 워크플로우 엔진에 접근하게 되면 하나의 워크플로우 엔진 인스턴스를 생성시킨다. 생성된 엔진 인스턴스는 클라이언트의 위치 정보를 클라이언트 모니터를 통해 넘겨받고 자신의 위치 정보를 클라이언트에게 알려준다. 이렇게 함으로써 클라이언트 모니터가 워크플로우 엔진 인스턴스를 생성한 후부터는 엔진 인스턴스와 클라이언트가 이벤트/요청 핸들러와 워크리스트 핸들러를 통해 1 대 1 로 대화를 하면서 업무를 처리하는 형태가 된다. 중앙 집중형 워크플로우 엔진의 생성과정은 [그림 2]와 같다.

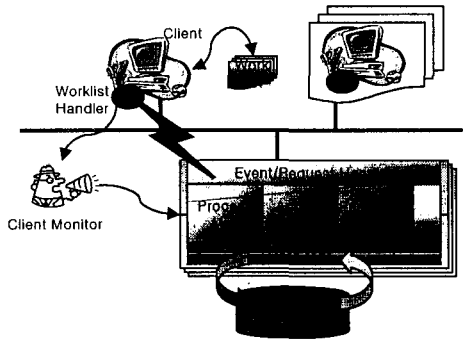

그림 2 중앙집중형 엔진의 생성

3.1.1.2 특징 및 장단점

이러한 형태의 워크플로우 엔진은 서버 시스템의 부하와 시스템등의 이상으로 엔진이 제대로 동작하지 못할 경우 워크플로우 업무 전체가 업무 마비될 수 있다는 문제점을 갖는다. 그러나 워크플로우 업무 흐름에 대한 모니터링이 용이하고 하나의 스케줄러를 갖는 워크플로우 엔진에 비해 병목(bottleneck)현상이 훨씬 적다는 장점을 갖는다. 또 하나의 시스템에 모든 엔진 인스턴스가 존재하므로 각 엔진간의 통신횟수가 잦을 경우에 유리하다.

3.1.2 분산 처리형 구조

중앙 집중형 워크플로우 엔진이 클라이언트 모니터에 로그인한 수만큼 서버내에 인스턴스가 생성된다. 반면에 제안하는 분산 처리형 워크플로우 엔진은 생성된 인스턴스를 정상적으로 로그인 과정을 마친 클라이언트에게 넘겨 줌으로써 클라이언트 머신에 서버 객체를 갖고 업무를 처리해 나가는 형태를 갖는다. 또 엔진과 클라이언트간의 인터페이스를 제공했던 이벤트/요청 처리기는 엔진간의 인터페이스를 제공함으로써 WfMC 에서 제안하는 인터페이스 4 의 역할을 수행한다.

3.1.2.1 분산 처리형 워크플로우 엔진의 생성

중앙 집중형 워크플로우 엔진과 마찬가지로 클라이언트 모니터는 서버에 데몬형태로 클라이언트의 로그인을 기다린다. 클라이언트가 로그인을 통해 클라이언트 모니터에 접근하게 되면 하나의 워크플로우 엔진 인스턴스를 생성하고 중앙 집중형 워크플로우 엔진과는 다르게 로그인을 한 클라이언트에게 그 엔진 인스턴스를 반환한다. 이렇게 되면 클라이언트는 반환된 워크플로우 엔진을 클라이언트 머신에 존재하는 하나의 객체처럼 접근 및 조작이 용이해진다. 분산 처리형 워크플로우 엔진의 생성과정은 [그림 3]와 같다.

3.1.2.2 특징 및 장단점

이러한 워크플로우 엔진은 특정한 서버라는 개념이 없이 완전히 분산된 형태를 갖는다. 따라서 어떤 클라이언트의 업무를 담당하는 워크플로우 엔진이 제대로 동작하지 않을 경우에도 다른 시스템에는 전혀 영향을 미치지 않는다. 그러나 클라이언트 시스템의 성능이 떨어질 경우에는 적용이 어렵다는 단점이 있다.

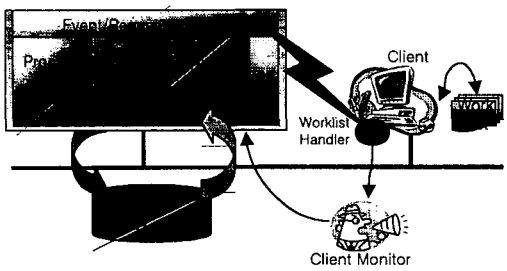

그림 3 분산처리형 엔진의 생성

3.2 서버 주도형 엔진

이 장에서는 워크플로우 엔진이 처리할 워크플로우 프로세스의 수만큼 엔진 인스턴스가 생성되는 형태의 엔진을 제안한다.

3.2.1 워크플로우 프로세스 처리 엔진의 생성

서버 주도형 엔진은 클라이언트 주도형 엔진과는 달리 클라이언트가 로그인 과정을 거칠때마다 새로운 엔진 인스턴스를 생성하지는 않는다. 클라이언트로부터 새로운 워크플로우 프로세스 수행을 요청을 받게 되면 클라이언트 모니터는 요청 받은 프로세스를 처리할 엔진 인스턴스를 하나 생성하게 된다. 이 후로 이 프로세스에 관련된 액티비티 인스턴스를 생성 및 할당, 수행자 선택, 필요한 데이터 접근 등의 모든 작업은 해당 프로세스 엔진에 의해 처리된다. 스케줄러에 의해 다음 수행될 액티비티를 임의의 수행자에게 할당할 때에는 현재 엔진의 위치 정보도 함께 넘겨주게 되고 클라이언트가 액티비티를 수행하는 과정 중에는 그 위치 정보를 이용해 엔진과 직접 통신을 함으로써 업무를 처리하게 된다. 워크플로우 프로세스 처리형 엔진의 생성과정은 [그림 4]와 같다.

3.2.2 특징 및 장단점

클라이언트 주도형 엔진은 워크플로우 흐름 정보가 필요할 때마다 데이터베이스에 접근을 해야 한다. 그러나 프로세스 처리형 엔진 인스턴스는 그 내부에 워크플로우 흐름 정보를 이미 가지고 있기 때문에 데이터베이스에 접근하는 횟수를 줄일 수 있다. 또한 수행되는 프로세스별로 엔진 인스턴스가 존재하기 때문에 스케줄링을 기다리는 등의 병목(bottleneck) 현상도 최소화 할 수 있고 업무 흐름에 대한 모니터링도 쉽다. 그러나 워크플로우가 장기간 수행되는 업무들의 집합이므로 각 클라이언트에서 대기 시간이 길면 길수록 워크플로우 엔진 인스턴스의 휴지 기간(idle time)이 늘어 서버에 부담을 주게 된다는 문제점이 있다. 워크플로우 엔진 인스턴스가 모두 서버측에 존재하기 때문에 중앙 집중형 서버가 갖는 문제점도 함께 갖는다.

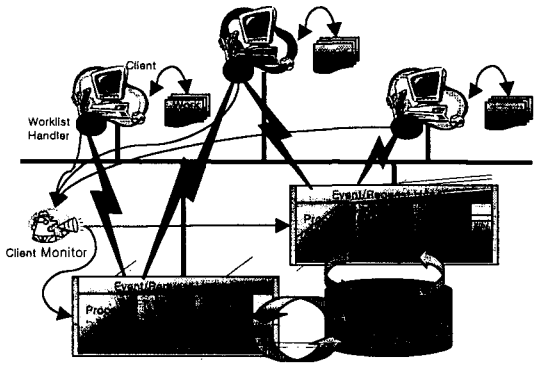

그림 4 워크플로우 프로세스 처리형 엔진의 생성

4. 결론 및 추후 연구과제

본 논문에서는 클라이언트 업무 처리형 엔진과 워크플로우 프로세스 처리형 엔진을 설계해 보았다. 그러나 3 가지의 엔진 형태가 업무 환경에 따라 각각 장단점이 있기 때문에 하나의 형태로 시스템을 구축하는 일은 쉬운 일이 아니다. 앞으로는 업무 환경에 따라 유연성있게 엔진의 형태를 변화시키면서 워크플로우 업무를 처리하는 워크플로우 시스템을 설계하도록 한다.

참고문헌

[1] John A. Miller, Amit P. Sheth, Krys J. Kochut and Xuzhong Wang, "CORBA-Based Run-Time Architectures for Workflow Management Systems," Journal of Database Management (JDM), Special Issue on Multidatabases, Vol. 7, No. 1 (Winter 1996). Idea Group Publishing.

[2] Karl R.P.H. Leung, Jojo M.L. Chung., "The Liaision Workflow Engine Architecture," Proceedings of the 32nd Hawaii International Conference on System Sciences, 1999.

[3] Mohan, C., "Recent Trends in Workflow Management Products, Standards and Research," In Proceedings of NATO Advanced Study Institute (ASI) on Workflow Management Systems and Interoperability, Istanbul, August 1997.

[4] 김동수, 배준수, 서영호, 허원창, 김영호, 강석호, "효율적 워크플로우 관리를 위한 시스템 특성 및 설계," 대한산업공학회/한국경영과학회 춘계공동학술대회 논문집, 1998.