클라이언트 성능을 고려한 적응적 워크플로우 시스템 구조

한상근* · 이도헌*

*전남대학교

An Adaptive Workflow System Architecture Considering the Client Capacity

Sang-Keun Han* · Doheon Lee*

*Chonnam National University

E-mail : skhan@dblab.chonnam.ac.kr

요 약

IT 산업의 급격한 발달에 따라 기업의 업무 환경도 빠르게 변화하고 있다. 그에 따라 활동성이 업무 처리의 중요한 요소로 부각되고, 이동 컴퓨팅 환경의 대중화를 앞당기고 있다. 기업 업무의 자동화를 담당하는 워크플로우는 이러한 변화에 민감하게 적응해야 한다. 따라서 본 논문에서는 이동 호스트를 포함한 모든 클라이언트가 업무에 참여하고 처리할 수 있도록 클라이언트의 성능에 따라 능동적으로 반응하는 워크플로우 시스템을 제안한다.

ABSTRACT

As developing IT industry, Enterprise environments are changed fast. As activeness is focused on an important factor in enterprise environment, mobile computing environment is being more popular. Workflow systems should be adapted to these changes because it takes change of automation of enterprise. So in this paper, I suppose an adaptive workflow system which covers all the types of client systems including mobile hosts and processes their works by capacity of each client.

Ⅰ. 서 론

워크플로우 엔진은 워크플로우의 런타임 실행 환경을 제공하고 워크플로우 프로세스 인스턴스의 처리와 수행을 관리하는 역할을 한다. 이러한 워크플로우 엔진은 IT 산업의 발달에 따라 그 응용 범위를 확대되어 왔다. 근래에는 기업의 요구에 적합한 이동 컴퓨팅 환경이 대중화되고 있다. 따라서 기업의 환경에 밀접한 관련을 맺는 워크플로우 시스템도 기존의 방식과는 다른 접근을 해야한다. 따라서 본 논문에서는 워크플로우 클라이언트 시스템의 성능에 따라 워크플로우 엔진 인스턴스를 성능이 좋은 서버나 클라이언트에 위치시키는 방법을 제시한다. 또한 병행되는 워크플로우 엔진 형태를 효과적으로 사용하기 위한 워크플로우 엔진의 최적화 방안에 대해서 제안한다.

Ⅱ. 워크플로우 시스템의 구성요소

1. 워크플로우 엔진

워크플로우 엔진은 다음과 같은 6개의 컴포넌트를 갖는다.

- 클라이언트 모니터(Client Monitor)
  워크플로우 클라이언트가 엔진에 접근하기 위해 로그인을 하는지 감시하고 있다가 로그인 되었을 때 워크플로우 엔진 인스턴스를 하나 생성하고 클라이언트 제어권을 새 워크플로우 인스턴스에게 넘겨준다.

- 프로세스 관리자(Process Manager)
  워크플로우 클라이언트로부터 워크플로우 수행 요청을 받고 워크플로우 업무 수행을 시작한다. 그리고, 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)
액티비티를 수행할 적당한 참여자를 찾아서 할당한다. 클라이언트가 갖는 업무의 양에 따라 적당한 참여자를 찾아내기도 함으로서 로드 밸런싱(load balancing)을 담당하기도 한다.

· 데이타 관리자 (Data Manager)
액티비티 수행에 필요한 문서나 데이터를 찾아서 할당한다. 데이터베이스 접근자의 역할도 함께하여 워크플로우 정보를 얻어낸다.

· 이벤트/요청 핸들러(Event/Request Handler)
워크플로우 클라이언트 측의 워크리스트 핸들러와 통신하면서 업무 처리 과정중에 발생하는 이벤트를 처리하기 위해 그 이벤트를 처리할 관리자 또는 클라이언트에게 이벤트 처리를 요청한다.

2. 워크플로우 클라이언트

워크플로우 클라이언트는 다음과 같은 2개의 컴포넌트를 갖는다.

· 워크리스트 핸들러(Worklist Handler)
클라이언트가 수행할 작업들에 대한 정보를 관리/유지하고 워크플로우 엔진과 대화를 하는 역할을 한다.

· 워크플로우 클라이언트(Workflow Client)
사용자와 워크플로우 엔진간의 대화 교류 매체로 사용자가 실제로 업무를 처리하는 도구이다.

3. 워크플로우 데이터베이스

프로세스 정의기에 의해 정의된 프로세스 정보를 비롯한 워크플로우 프로세스의 구조, 액티비티, 액티비티의 흐름, 수행자 정보등 워크플로우 수행에 필요한 모든 정보를 갖고 있다. 필요한 경우에 각 정보를 갖는 데이터베이스는 서로 다른 시스템에 존재할 수 있다.

4. 워크플로우 엔진의 동작과정

워크플로우 엔진은 [그림1]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 핸들러에게 액티비티를 수행할 준비가 되어 있음을 알리고 과정 ⑨에서 이벤트/요청 핸들러는 과정 ④를 통해 얻어진 수행자에게 처리해야할 새로운 액티비티가 할당되었음을 알린다. 위의 과정 ④와 ⑥에서 각각의 수행자와 데이터가 유효하지 않을 경우 워크플로우 업무 수행은 실패한다.

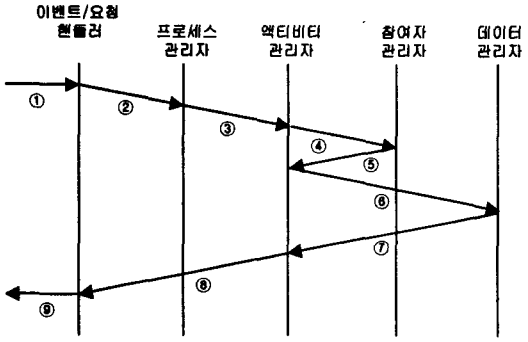

그림 1. 워크플로우 엔진의 동작과정

III. 워크플로우 엔진

본 논문에서는 두 가지 형태의 워크플로우 엔진을 사용한다. 고성능의 서버에서 쓰레드를 이용해 워크플로우 업무를 처리하는 중앙 집중형 워크플로우 엔진 형태는 주로 클라이언트가 성능이 떨어질 때 사용한다. 반면에 분산 처리형 워크플로우 엔진은 클라이언트로 사용되는 개인 시스템의 처리 능력을 이용해 과중한 워크플로우 엔진의 업무를 분담한다.

1. 분산 처리형 워크플로우 엔진

클라이언트 모니터가 서버에 데몬(Deamon) 형태로 클라이언트의 로그인을 대기하고 클라이언

트가 로그인을 마치면 워크플로우 엔진 인스턴스를 하나 생성한다. 그런 다음, 워크플로우 엔진 인스턴스를 클라이언트에게 반환해 준다. 이렇게 하면 클라이언트는 반환된 워크플로우 엔진을 로컬 시스템에 존재하는 하나의 객체처럼 접근하고 조작이 가능해진다. 이 때 클라이언트 모니터는 각 워크플로우 엔진 인스턴스를 가지고 있는 클라이언트의 위치 정보를 클라이언트 목록에 등록하게 된다. 이 목록에 등록된 위치 정보들은 필요할 때 다른 위치에 있는 워크플로우 엔진과 상호 대화하는데 사용된다.

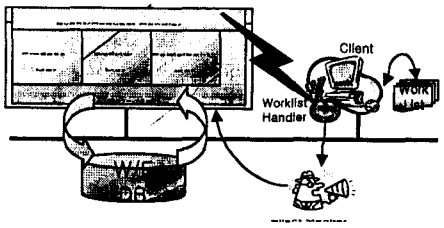

그림 2. 분산처리형 워크플로우 엔진

2. 중앙 집중형 워크플로우 엔진

중앙 집중형 엔진 형태는 두가지 방식을 갖는다. 첫 번째는, 분산 처리형 엔진과 마찬가지로 클라이언트가 로그인을 통해 클라이언트 모니터에 접근하면 하나의 워크플로우 엔진 인스턴스가 발생한다. 워크플로우 엔진 인스턴스는 클라이언트 모니터의 엔진 인스턴스 목록에 등록된 후부터 클라이언트와 통신을 통해 워크플로우 업무를 처리하게 된다. 이 때, 엔진 인스턴스는 클라이언트 모니터로부터 클라이언트의 위치 정보를 얻어내고 인스턴스 생성이 완료되었을 때에는 그 위치정보를 이용해 클라이언트에게 등록되었음을 알림과 동시에 자신의 위치정보도 알려준다. 이렇게 함으로써 클라이언트 모니터가 워크플로우 엔진 인스턴스를 생성한 후 엔진과 클라이언트가 1대 1로 대화를 하면서 업무를 처리하는 형태가 된다.

상대적으로 워크플로우 프로세스와 액티비티의 수행 요구시간이 짧을 경우에는 프로세스에 대응하는 엔진 인스턴스를 만드는 것이 유리하다. 따라서 두 번째 방식은, 클라이언트가 로그인할 때 인스턴스가 생성되는게 아니라 하나의 프로세스 요청이 들어 왔을 때 그 프로세스를 전담하는 엔진 인스턴스가 생성되는 것이다.

그러나 위와 같은 두가지 형태의 중앙 집중형 워크플로우 엔진은 실질적으로 효과적이지 않다. 각각 클라이언트와 수행중인 프로세스를 담당하는 수행자가 작업중이지 않을 경우, 또는 실제로 작업을 완료하거나 엔진과 통신을 통해 업무를 처리하지 않을 때에는 서버내에 존재하는 엔진 인스턴스는 휴지시간(idle-time)을 갖는다. 클라이언트의 수 혹은 요청된 워크플로우 프로세스의 수만큼 엔진 인스턴스가 생성되었으나 클라이언트와의 통신이 없다면 서버의 리소스를 낭비함으로써 서버에게 많은 부담을 주게 되는 것이다.

따라서, 서버를 효율적으로 사용하기 위해서는 클라이언트나 프로세스의 수만큼 엔진 인스턴스를 생성시키지 않고 적당한 수의 인스턴스가 여러 클라이언트나 프로세스를 처리하는 것이 필요하다.

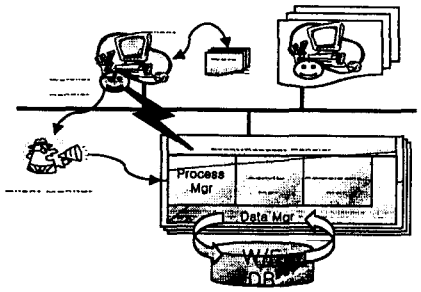

그림 3. 중앙집중형 워크플로우 엔진

IV. 워크플로우 엔진의 적응적 배치

근래의 클라이언트 시스템은 개인용 컴퓨터뿐만 아니라 휴대용 컴퓨터, 개인정보단말기(PDA), 스마트폰, 차세대 이동통신 단말기로 꼽히고 있는 IMT-2000 등의 이동 컴퓨팅환경이 대중화되어 가고 있다. 이러한 클라이언트 환경의 변화는 워크플로우 환경에도 큰 영향을 미친다. 원하는 순간에 정보를 얻을 수 있을 뿐 아니라 장소에 구애받지 않고 자신의 업무를 어디에서든지 실시간으로 처리할 수 있다는 장점이 기업의 업무환경에 적합하기 때문이다. 그러나 아직까지는 케이블을 이용한 네트워크 접근과 비교하여 속도가 떨어지고 개인용 컴퓨터에 비해 성능이 훨씬 떨어진다는 문제점이 있다. 따라서 이동 컴퓨팅 환경에서 워크플로우 업무를 수행하기 위해서는 기존의 클라이언트와는 다른 방법으로 접근 해야한다.

1. 워크플로우 엔진의 적응적 배치 방안

워크플로우 엔진 의 적응적 배치 목적은 이동통신 단말기의 성능을 고려하여 최대한 클라이언트가 갖는 시스템상의 부담을 줄이고 서버와의 통신을 통한 업무 처리를 주로 담당할 수 있도록 하는 것이다. 따라서 클라이언트의 성능에 따라 엔진의 형태를 적절하게 생성시키는 과정이 필요

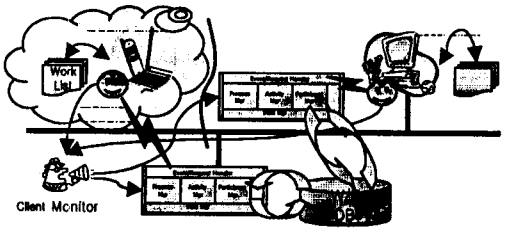

그림 4. 클라이언트 성능에 따른 엔진의 적응적 배치

하다. [그림4]는 클라이언트의 성능에 따라 워크플로우 엔진 인스턴스를 서버 측 혹은 클라이언트 측에 두고 업무를 처리하는 모습을 보이고 있다. 워크플로우 엔진의 위치를 결정하는 클라이언트의 성능은 절대적인 기준으로 구분할 수 있는 것이 아니므로 워크플로우 엔진이 요구하는 시스템 성능을 기준으로 구분하도록 한다. 이러한 과정은 클라이언트 시스템의 시스템 콜을 이용해 그 성능을 알아낸 후 워크플로우 엔진 인스턴스를 반환 받아 처리가 가능한 지의 여부를 클라이언트 모니터가 판단하고 처리한다. 또는 클라이언트 애플리케이션에서 어느 엔진형태를 사용할 것인지 옵션을 통해 선택할 수 있는 경우도 가능하다.

2. 워크플로우 엔진의 최적화 방안

클라이언트의 성능에 따라 엔진의 인스턴스를 생성하는 방법만으로는 워크플로우 엔진이 최적화되거나 때에 따라 성능이 향상되지는 않는다. 클라이언트의 수 혹은 프로세스의 수만큼 증가하는 엔진 인스턴스가 서버에 부담을 주기 때문이다. 또, 서버 측에 존재하는 엔진 인스턴스는 항상 액티비티를 처리하는 것이 아니다. 보통의 경우, 서버의 메모리상에 존재하고 있다가 자신이 담당하는 클라이언트 혹은 프로세스 수행자와 통신을 할 때에만 엔진의 역할을 한다. 이러한 특성은 웹서버가 갖는 특성과 비슷하다. 따라서 웹서버의 업무 요청에 따른 서버의 동작 형태를 살펴보면 쉽게 효율적인 워크플로우 엔진의 생성 형태를 연구할 수 있다.

가장 널리 사용되고 있는 웹서버인 아파치의 설정파일을 살펴보면 다음과 같은 성능향상을 위한 솔루션을 찾을 수 있다. 웹서버에서는 각 클라이언트의 요청의 수에 따라 웹서버 프로세스를 생성하고 제거하는 것이 아니라 서버에 생성된 서버 프로세스를 적극 재활용함으로써 웹서버의 성능을 향상시키는 것을 알 수 있다. 따라서, 워크플로우 엔진도 위와 같은 설정을 함으로써 훨씬 효율적인 형태를 갖출 수 있을 것이다. 즉, 아파치의 KeepAlive 옵션과 같이 임의의 워크플로우 엔진 인스턴스가 하나의 워크플로우 프로세스를 처리했다고 해서 그 엔진 인스턴스를 제거하는 것이 아니라 다른 워크플로우 프로세스 수행을 위해 그냥 두는 것이다. 또, MinSpareServer, MaxSpareServer 옵션과 같이 스레드화된 엔진 인스턴스의 작업량에 따라 생성할 수 있는 최대 엔진 인스턴스의 수와 최소 엔진 인스턴스를 정의함으로써 원활한 클라이언트의 업무 처리를 유지한다. 마찬가지로, 워크플로우 시스템을 실행시킬 때 생성될 워크플로우 엔진 인스턴스의 수를 설정할 수도 있다. 최대한 받아들일 클라이언트의 수도 정의할 수 있는데, 이렇게 하면 한 시스템으로 업무 처리가 편중되는 현상을 막음으로써 로드 밸런싱을 유지할 수 있다.

V. 결 론

본 논문에서는 클라이언트의 성능에 따라 엔진이 능동적으로 배치되는 형태의 워크플로우 엔진을 제안하고, 제안한 엔진의 능력을 최대화하기 위해 엔진을 최적화하는 방안을 제시하였다. 향후에는 제안한 워크플로우 시스템을 실제로 구현하고 그 성능을 평가함으로써 다른 워크플로우 시스템과 비교를 통해 더욱 효과적인 워크플로우 시스템을 설계하도록 한다.

감사의 글

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

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] 한상근, 이도헌 "워크플로우 시스템의 프로세스 구조 대안 비교", 정보처리학회 추계학술대회 논문집, 2000.