P2P 기반 분산 워크플로우 프레임워크

이이섭[O*] 박수현** 백두권*
*고려대학교 컴퓨터학과 **국민대학교 정보관리학부
*{eesub, baik}@software.korea.ac.kr *shpark21@kookmin.ac.kr

Distributed Workflow Framework based on Peer to Peer

Leesub Lee[O*], SooHyun Park**, Dookwon Paik*
*Dept. of Computer, Korea University, **School of Business IT, Kookmin University

요 약

본 논문에서는 기존의 워크플로우 표준 및 구현방법에 대한 장단점을 정리하고 최근에 각광을 받고 있는 P2P기술을 기반으로 하는 워크플로우 시스템에 대하여 제안하고자 한다. P2P기술은 각 피어의 자원을 최대로 활용하여 서버의 부하를 줄여주는 장점을 갖고 있었으나, 파일 공유 정도의 단순한 상호 작용 기능만 제공되고 있다. 복잡한 형태의 상호 작용을 요구하는 워크플로우를 지원할 수 있는 구조를 제시함으로서, C/S, CORBA, HTTP 보다 완벽하게 분산된 구조의 워크플로우 시스템을 구축할 수 있게 되었다. 이러한 워크플로우은 보다 확장성 있고 견고하고 고성능을 제공하게 된다. 또한 본 연구에서는 확장성, 성능, 그리고 메모리 요구량에 대하여 기존 시스템과 비교하였다.

1. 서 론

워크플로우 시스템은 응용프로그램에서 프로세스를 독립시켜 프로세스 독립성을 제공함으로써 다양한 경영 환경 변화에 부합되는 프로세스의 변혁을 보다 적극적으로 지원한다. 초기의 워크플로우 연구는 BPR을 구현하는 개발 도구로 시작하여 EAI, SCM, B2Bi 등 프로세스가 필요로 하는 모든 응용과 솔루션에 적용되어 사용되고 있다.

그러나 최근의 연구 되고 있는 분산 기술 구조 즉 웹 서비스, 그리드 컴퓨팅, 에이젼트 등을 살펴보면 가장 기본적인 서비스를 제공할 뿐 워크플로우가 제공하고 있는 프로세스 관리와 같은 고도화된 서비스를 요구하고 있다. 또한 이러한 기술 구조에서는 단순한 라우팅 기능뿐 만 아니라 분산성, 확장성, 독립성, 결함 허용성, 단순성 등을 요구한다. 이를 위하여 본 논문에서는 완전 분산 구조의 워크플로우 시스템을 P2P기반으로 제시하고자 한다.

2장에서는 관련되는 워크플로우 연구를 살펴보고 3장에서는 P2P기반의 워크플로우 시스템의 구조와 원리를 살펴보고 사용 시나리오별로 설명한다. 4장에서는 확장성, 성능, 그리고 메모리 요구량에 대하여 기존시스템과 비교하였다.

2. 관련 연구

WfMC 참조 모델[1]은 구현 기술과는 독립성을 갖게 제정되었으며 이후에 연구되거나 개발된 시스템들의 기본 모델을 제공하였다. 타 워크플로우와의 연동을 위한 Interface 4를 표준화 하였으며 시범적으로 MIME를 기반으로 구현되었다[2]. 이는 메시징 백본을 사용하여, 신뢰성이 부족하고, 지연시간을 예측하기 힘들며, 오류 발생시 처리가 매우 복잡해지는 문제가 있을 뿐만 아니라 구현이 너무 복잡하여 프로토타입 정도만 개발되었다.

OMG에서는 분산 객체 모델인 CORBA를 표준으로 jFLOW[3]를 제안하였다. 워크플로우에 대한 연구가 가속화되자 자신의 분산 객체 환경에 작업 흐름 제어 기능을 제공하기 위한 것이었다. 이를 위해서는 기존에 만들어진 워크플로우 엔진을 컴포넌트 별로 완전히 CORBA를 기반으로 완전히 재개발해야하는 어려움이 발생하게 된다. 인터넷 기반의 서비스 제공을 위하여 HTTP를 기반으로 하는 시스템으로 SWAP[4]이 제안되었다. 이는 단지 HTTP의 확장에 대한 제안이며 표준인 WfMC에서 갖추어야 할 요구사항이 모두 정의 된 것은 아니다. 최근에는 SWAP을 기반으로 하여 워크플로우 엔진 간의 메시지에 대한 표준 포맷으로 Wf-XML이 제안되었다.[5]

비록 SWAP을 따르지 않았지만 HTTP기반으로 구현된 대표적인 시스템으로써 WebWork[6]가 있다. 워크플로우에 대한 접근 방법을 브라우저에서 해당 CGI 프로그램의 호출로 처리하는 방법이다. 방법상으로는 웹에 기반한 워크플로우 시스템으로 볼 수는 있겠지만, 실제적으로 브라우저는 단지 Thin Client 역할만하며 서버가 모든 역할을 맡기 때문에 부하가 서버로 집중되게 된다.

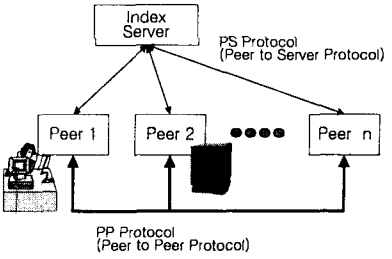
일반적으로 HTTP를 기반으로 하는 시스템은 No Event Back channel, Routing and Naming Convention ,Non-ubiquity의 제약을 갖고 있기 때문에 Magi[7]에서는 HTTP, WebDev, CPAM 등 최신의 인터넷 표준들을 종합하여 문제를 해결하고자 하였다. 복합적인 인터넷 표준 프로토콜을 연동한 서버를 각각의 클라이언트에 내장시켜 비연결성 서비스를 제공하여 주었다. 그러나 데이터의 동기화 문제로 중앙 집중식 HTTP 서버와 객체 저장소를 통해서 모든 데이터의 전송이 이루어지므로 서버에 부하가 집중된다. 또한[8]에서의 실패와 같이 여러 가지 프로토콜을 조합하여 다른 용도의 프로토콜을 구현하려는 시도는 많은 한계를 갖게 된다.

3. P2P기반의 워크플로우 시스템

본 논문에서 P2P 기술에 관심을 갖는 이유는 이 기술이 단순하면서도 확장성이 매우 높고 HTTP가 보안상의 이유로 갖게 된 기능상의 제약이나 지역 자원의 적극적 활용이 가능하기 때문이다[9]. 또한 사용자가 네트워크에 항상 연결되지 않아도 되는 상황을 허용하는 비연결성 서비스를 제공한다는 점이다. 비연결성은 서버와 같이 안정된 자원이 아닌 일반 사용자의 자원을 활용하는 경우 필수적이다.

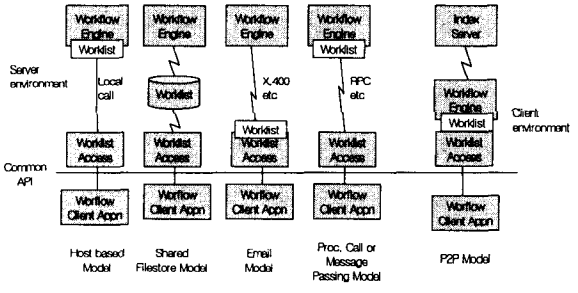
P2P 기반 워크플로우는 [그림 1]과같이 인덱스 서버와 여러 개의 피어로 구성되는 전형적인 P2P구조를 갖고 있다. 기존의 워크플로우가 라우팅 및 데이터관리를 모두 서버에서 처리하는 것과는 달리 각 피어에서 상당 부분을 처리한다. 각 피어는 수작업의 경우는 사용자가 자동화된 작업의 경우는 응용프로그램이 처리한다. 전역적인

정보의 처리는 PS Protocol로 이루어지며, 실제 작업에 필요한 정보 처리는 PP Protocol로 이루어진다. 중앙집중식 모델에서는 각 클라이언트간의 통신은 모두 서버를 거치게 되지만, PP프로토콜 사용을 극대화함으로서 성능을 향상시킨다.



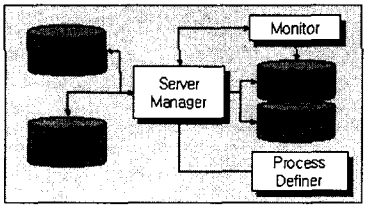
[그림 1] P2P 기반 워크플로우의 전체 구성

P2P 기반 워크플로우는 단순 파일 공유보다는 더 복잡한 다중 계층의 구조를 갖게 된다. [그림 2]는 WfMC[1]에서 서버와 클라이언트의 컴포넌트 상의 배치 방법에 P2P기반 워크플로우를 추가하여 다양한 구성 방법을 분류 비교하였다. 이로서 엔진, 워크리스트, 워크리스트 핸들러, 클라이언트 순의 통신 계층이 있음을 보여주며, 각 모듈의 배치에 따라 분류가 됨을 알 수 있다.



[그림 2] 컴포넌트의 배치에 따른 분류

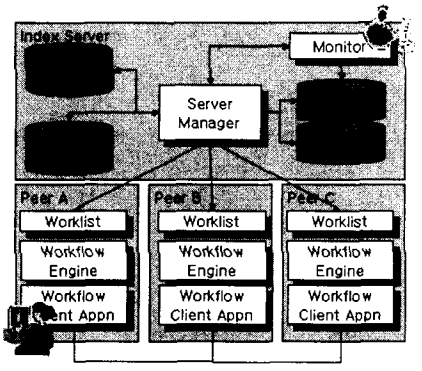
각 피어는 [그림 2]와 같이 완벽한 최소규모의 워크플로우 시스템을 구성하고 있다. 기존의 워크플로우 시스템이 서버에 모든 사용자의 프로세스에 대한 정보를 저장하고 처리하는 것과는 달리 여기에서는 해당 피어가 참여하고 있는 프로세스 정보만을 저장하고 처리한다. 즉 기존의 모든 워크플로우 시스템은 엔진과 사용자가 1:n의 관계를 갖지만 P2P기반 워크플로우의 경우에는 1:1 관계를 갖는다. 사용자는 각 피어의 워크리스트에서 자신에게 할당된 워크 아이템을 가져와 업무를 처리한다. 처리 후 라우팅은 자신이 갖고 있는 워크플로우 엔진에 의하여 결정되며 결정된 다음 작업의 피어에게 전달된다.



[그림 3] 인덱스 서버의 구성

인덱스 서버에서 가장 중요한 역할을 맡고 있는 워크그룹 저장소에는 각 프로세스 인스탄스에 대하여 참여하고 있는 피어들의 ID 리스트를 저장하고 있는데, 라우팅, 모니터링, 복구 등 전역적인 용도로 사용된다. 실제 인스탄스 데이터는 참여하고 있는 각 피어에 중복되어 저장된다. 자원 저장소는 워크플로우에서 사용자 및 서비스 디렉터리의 역할을 수행한다. 모니터는 전체적인 또는 다른 피어와 연관되는 객체들의 상태 정보를 관찰하기 위하여 사용된다.

프로세스의 정의는 서버와 연결된 프로세스 정의기에 의하여 수행된다. 프로세스 정의 시에는 작업에 할당될 자원을 결정해야 하기 때문에 자원 저장소를 참조하며, 서브 프로세스를 사용할 때와 완성된 프로세스 정의를 저장하기 위하여 프로세스 저장소를 참조한다. 기존의 워크플로우 시스템에서의 프로세스 정의와 동일하다.



[그림 4] P2P기반 워크플로우의 구조

예를 들어 피어 A에서 새로운 프로세스 객체를 생성시키는 경우에는 서버에서 자신이 생성시킬 수 있는 프로세스 정의 리스트에서 프로세스를 선택한다. 피어 A에서는 서버에서 전송 받은 프로세스 정의를 참조하여 인스탄스 정보를 생성하여 피어 A에 초기화 시키고, 사용자에게 필요한 데이터를 입력 받아 다음 라우팅을 계산한다. 도출된 다음 피어들을 서버의 작업그룹에 등록 시키고 다음 피어들에게 인스탄스 정보를 전송한다.

다음 피어가 B인 경우 사용자는 워크리스트에서 피어 A로부터 작업이 도착했음을 알 수 있다. 사용자가 작업을 완료하면 각 피어는 다음 라우팅을 계산하고 진행 시킨다. 그리고 서버의 워크 그룹을 갱신하고 필요시 관련된 피어들의 인스탄스 데이터를 동기화 시킨다.

만약 피어 C에서 작업이 완료되고 프로세스 정의에 따라 프로세스를 종료시키게 되는 경우에는 서버의 워크그룹을 삭제 시키고 각 피어들에게 완료되었음을 알린다. 각 피어들의 작업항목을 사용자가 명시적으로 삭제하지 않는 한 계속 유지가 된다.

사용자가 본인이 참여하는 인스탄스에 대한 모니터링은 각 피어에서 지역적 처리가 가능하다. 기존의 워크플로우에서는 이러한 형태의 모니터링이 서버에 상당한 부하를 주게 된다. 시스템 관리자나 작업 관리자가 자신이 소속되어 있지 않은 작업들의 수행 현황을 살펴보거나 업무 조정을 하는 경우가 있다. 이것은 발생 빈도는 낮으나 비용이 높은 경우이다. 이 경우에는 작업그룹 저장소를 참조하여 각 피어에게 상태 정보를 동적으로 수집하는 방법을 생각할 수 있다. 이 방법은 상태 정보 수집 시에 시간 지연이 있으며, 확률은 낮지만 특정 프로세스 객체의 참여자가 모두 비연결 상태에 있을 때 작업 현황을 수집하지 못하는 경우가 있다. 다른 방법으로 서버에 로그 데이터베이스를 유지하는 방법이 있는데, 이 때는 각 피어가 작업을 수행할 때 마다 작업 아이템을 로그 데이터베이스에 기록해야 한다. 이 방법은 상태 정보 수집에 대한 시간지연이 없고 항상 작업 현황을 수집하여 보여줄 수 있으나, 모든 작업 시에 서버에 작업 아

이템을 기록해야 하므로 서버에 부하를 줄 수 있다.

시스템 장애가 발생하는 경우는 피어 장애와 서버 장애로 분류된다. 예를 들어 피어 B 장애 시 복구의 경우에는 서버의 워크그룹 저장소에서 피어 B가 소속된 모든 워크그룹을 전송 받고, 각 워크그룹에 소속된 피어들로부터 작업 항목을 전송 받음으로써 복구 한다. 이 경우 한 워크그룹 내에 있는 모든 피어들이 동시에 파괴되지 않는 한 복구가 가능하다.

서버에 장애가 있는 경우 자원 저장소와 프로세스 저장소는 정적인 데이터를 저장하고 있으므로 용이하게 백업할 수 있고 복구가 가능하다. 동적인 데이터를 저장하는 작업그룹 저장소를 복구하려면 각 피어들에게 작업그룹을 동적으로 수집함으로써 복구 할 수 있다. 등록된 피어들의 리스트가 없으면 복구가 불가능하나 이는 매우 정적이고, 용량이 적으므로 용이하게 관리될 수 있다.

4. 기존 시스템과의 비교

[표1]은 가장 많이 발생되는 작업별로 기존의 중앙집중식 워크플로우와 P2P 기반 워크플로우의 부하를 정리하였다. 중앙집중식 시스템의 경우에는 목록 보기, 상태 정보 보기와 같은 워크리스트 접근에 대한 부하가 매우 크다. 이러한 작업들은 P2P의 경우 자신에 관련된 워크리스트가 모두 지역적으로 저장되고 관리되어짐으로 서버에 전혀 부하를 주지 않는다.

O(Login 수)	없음	O(Login 수)
O(조회수)	없음	O(조회수)
O(인스탄스 수 * 평균 작업 수)	O(인스탄스 수 * 평균 작업 수)	O(인스탄스 수 * 평균 작업 수 * 평균 관련 피어 수)
O(인스탄스 수 * 평균 작업 수 * 평균 전송량)	없음	O(인스탄스 수 * 평균 작업 수 * 평균 관련 피어 수)

[표1] 서버에 대한 부하 비교

작업 처리의 경우에는 같은 빅 O 표기에서는 같은 부하를 보여주지만 서버의 워크그룹 갱신 정도의 작업만 처리하기 때문에 실제로는 서버의 부하가 매우 적다. 또한 상태정보 동기화 및 자료 전송 작업은 각 피어들 사이에 자료의 동기화가 필요하기 때문에 '평균 관련 피어 수' 배 만큼의 피어간 데이터 전송량이 더 발생되게 된다. 데이터 전송량에 대한 서버의 오버헤드의 중요성과 처리에 대해서는 [11]에서 언급되었다. 그러나 서버와의 통신이 없이 피어 사이에서 처리하게 되므로 서버에는 부하가 거의 없게 된다.

O(정의된 프로세스의 개수)	O(정의된 프로세스의 개수)	없음
O (인스탄스 수)	없음	O(인스탄스 수 * 평균 작업 개수)

[표2] 서버에 대한 저장 공간 비교

저장 공간의 측면에서 기존의 경우 한 서버에 비즈니스 프로세스 객체의 상태 정보가 모두 저장된다. 그러나 P2P의 경우에는 각 피어에 관련이 되는 모든 상태정보가 중복 저장된다. 따라서 P2P의 경우 각 피어에서는 기존의 경우보다 인스탄스의 평균 작업 수 배 만큼의 디스크 용량이 더 필요하다. 이러한 상태정보는 매우 저렴한 클라이언트에 저장되므로 상대적으로 적은 비용을 요구할 뿐만 아니라, 고가의 서버에 상태 정보가 저장될 필요가 없다. 또한 중복 저장되는 상태 정보는 처리 성능을 높여 주며, 유사시 복구용 데이터로 사용된다.

워크플로우의 용량의 증대에 대한 병목 지점은 서버이다. 부하와 저장 공간의 비교에서 살펴 본 바와 같이 P2P 구조를 갖는 워크플로우 엔진의 특징이 서버의 부하와 저장 공간을 급격하게 감소시키기 때문에 고가의 서버 자원의 요구가 감소되어 확장성에서 매우 유리하다.

5. 결론 및 향후 연구 과제

P2P는 매우 새로운 패러다임이며 파일 공유와 같은 일부 분야에서 매우 성공적이었다. 본 논문에서는 이러한 P2P 기술을 워크플로우에 적용함으로써 다수의 사용자를 지원하고 저렴한 지역 자원을 활용하여 비용을 절감시키는 구조를 제안하였다. P2P기반의 워크플로우 시스템의 구성과 동작 알고리즘을 제시하였으며, 특히 메모리 사용량, 서버의 등으로 기존 시스템과의 확장성에 대하여 비교하였다.

향후과제로는 P2P 기반 워크플로우 시스템의 구조와 PS 및 PP 프로토콜을 Z Notation과 같은 형식적인 방법으로 모델링하고 이 프로세스의 생성, 처리, 완료, 모니터링 및 복구 등에 대하여 검증을 수행하고 난 후 프로토타입을 구현해 보는 것이다. 또한 중앙 집중식으로 된 가장 큰이유가 모니터링 기능 구현의 용이성이므로 모니터링에 대한 분산 구조의 설계와 분석 및 검증을 보다 세밀하게 연구해 볼 필요가 있다. 특히 개방되어져 있는 분산 환경 하에서의 보안 모델과 트랜잭션 처리 모델을 설정하여 검증해 볼 필요도 있다. 모델의 타당성을 지원하기 위하여 각종 분산 모델 별로 성능에 대한 시뮬레이션이 필요할 것이다.

6. 참고 문헌

[1] WfMC: "Workflow Reference Model", document number WfMC-TC-1003 version 1.1, January 1995, http://www.aiim.org/wfmc/standards/docs/tc003v11.pdf

[2] Steve Silverberg, Mordechai Beizer "Microsoft MAPI Workflow Framework Concepts and Facilities" Microsoft Corporation Wang Laboratories, Inc. February 21, 1996

[3] OMG: "Workflow Management Facility" April 2000, ftp://ftp.omg.org/pub/docs/formal/00-05-02.pdf

[4] Swenson, Keith: Simple Workflow Access Protocol (SWAP) Internet Draft (work in progress). (available from http://www.ics.uci.edu/~ietfswap/SWAP9807.html).

[5] Workflow Management Coalition: Workflow Standard Interoperability - Wf-XML Binding. Document Number WfMC-TC-1023. Draft 0.9.2. (available from http://www.wfmc.org)

[6] Mill97a Miller J.A, Palaniswami D., Sheth A., Kochut K., Singh H.: 'WebWork:METEOR2's Web-based Workflow Management System", Technical Report UGACS-TR-97-002, University of Georgia, Department of Computer Science

[7] Gregory Alan Bolcer,"Magi: An Architecture for Mobile and Disconnected Workflow" Endeavors Technology Inc. 46 MAY JUNE 2000 HTTP://computer.org/internet/

[8] R. Khare, "Building the Perfect Beast: Dreams of a Grand Unified Protocol," IEEE Internet Computing, vol. 3, no. 2,Mar.-Apr. 1999, pp. 89-93.

[9] Oram, Andy,"Peer-to-Peer: Harnessing the Power of Disruptive Technologies",O'Reilly, 2001

[10] F. Leymann and W. Altenhuber "Managing Business Processes as an Information Resource." IBM Systems Journal, 33(2):326-348, 1994.