인터넷 규모의 협업 지원을 위한 워크플로우 수행 모델

(A Workflow Enactment Model supporting Internet-scale Collaboration)

이 이 섭 † 박 수 현 †† 백 두 권 †††
(Lee-Sub Lee) (Soo-Hyun Park) (Doo-Kwon Baik)

요 약 이 논문에서는 웹 서비스와 그리드 컴퓨팅과 같은 차세대 인터넷 서비스 환경에서 프로세스 관리 기능을 제공하기 위한 워크플로우 수행 모델을 제안하였다. 기업 환경을 위한 기존 워크플로우 수행 모델은 이러한 환경의 확장성과 신뢰성에 대한 요구를 만족시키지 못하고 있다. 이러한 요구사항을 만족시키기 위하여, 이 연구에서는 P2P 모델을 워크플로우 수행 모델에 적용하였다. 신뢰성 지원을 위한 복구기법을 집합론을 사용하여 수학적으로 기술하고 증명하였고, 시뮬레이션을 통하여 같은 자원을 사용하는 중앙 집중식 모델에 비하여 약 100배 이상의 신속한 응답시간 향상을 보임을 알 수 있었다.

키워드 : 워크플로우 시스템, P2P 시스템, 웹 서비스, 그리드 컴퓨팅

Abstract This paper proposed a workflow enactment model to provide process management functions for the next generation Internet service environment such as Web Service and Grid Computing. The existing enterprise workflow enactment models could not meet the scalability and reliability requirements for aforementioned environments. To satisfy these requirements, this study addressed on applying P2P model to the workflow enactment model. Mathematical description and proof were employed to show the recovery based on set theory and we also showed performance enhancement through simulations, which demonstrated a comprehensive result giving more than 100 times fast response time than the existing centralized model on the same resources.

Key words : Workflow System, Peer-to-Peer System, Web Service, Grid Computing

1. 서 론

최근에 차세대 인터넷 서비스로 웹 서비스, 그리드 컴퓨팅 등에 대한 연구가 활발하게 진행되고 있다. 웹 서비스란 인터넷에서 사용 가능하고, 특정 운영체계와 언어에 종속되지 않으며, 표준 XML 메시지 시스템을 사용하는 서비스를 말한다[1]. 그리드란 동적으로 구성되는 가상 조직들이 협력하여 자원을 공유하고 문제를 해결하는 기반 구조이다[2]. 즉, 지리적으로 분산되어 있는 고성능 컴퓨팅 자원을 인터넷에서 상호 연동하여, 조직과 지역에 관계없이 사용할 수 있는 환경이다. 기존의 웹은 자료를 공유하는 정도의 서비스만을 제공하였는데, 단순자료의 공유 뿐 아니라 프로세서, 스토리지 등 다양한 자원을 공유하여, 고성능 컴퓨팅을 제공하는 것이 그리드이다. 이 서비스들의 가장 중요한 공통점은 서비스되는 환경이 하나 또는 몇 개의 조직들이 아닌 인터넷 전체를 대상하고 있다는 점이다. 이 서비스들에 대한 기본적인 표준들이 구체적으로 논의가 되고, 정리가 마무리됨에 따라, 워크플로우 기술을 중심으로 하는 협업 기능에 대한 연구가 새롭게 진행되고 있다[3,4].

워크플로우 관리 시스템은 다수의 상이한 정보 자원의 통합과 비즈니스 프로세스의 전반적인 수행 상황을 모니터링하기 위한 가장 효과적이고 현실적인 도구이다[5]. 워크플로우는 초기에 BPR(Business Process Reengineering)을 지원하기 위한 비즈니스 프로세스 자동화 도구로서 시작되었지만, 그룹웨어, ERP(Enterprise Resource Planning), EAI(Enterprise Application Integration), BPMS(Business Process Management

† 정 회 원 : 삼성SDS 연구원
eesub@lycos.co.kr
†† 정 회 원 : 국민대학교 비즈니스IT학부 교수
shpark21@kookmin.ac.kr
††† 종신회원 : 고려대학교 컴퓨터학과 교수
baik@dblab.korea.ac.kr
논문접수 : 2003년 8월 28일
심사완료 : 2003년 11월 6일

System), ebXML, 그리고 e-commerce 등의 분산 컴퓨팅의 협업을 위한 핵심 기술로서 자리 잡고 있다. 워크플로우가 여러 가지 많은 분야에서 적용되고 있기 때문에 여러 가지 형태의 표준들이 제정되었고, 새로운 분야에 대한 다양한 표준 제정 노력이 계속되고 있다.

최근에 그리드 컴퓨팅과 웹 서비스와 같은 차세대 인터넷 서비스들에 대하여, 진보된 워크플로우 기능을 제공하기 위한 연구로서 WSFL(Web Service Flow Language)[3], GSFL(Grid Service Flow Language)[4]과 같은 연구들이 진행되고 있다. 관련된 주요 연구들은 프로세스 정의를 공유하기 위한 프로세스 정의 모델의 표준화에 집중되었으며, 실행 모델에 대한 연구는 매우 미진한 상태이다. 차세대 인터넷 서비스들의 운영 환경에서 프로세스 실행 모델에 영향을 미치는 주요 요구사항들을 살펴보면 아래와 같다.

- 확장성 : 현재까지의 전통적인 분산 워크플로우 기술들은 단일 서버 또는 다중 서버가 하나 또는 몇 개의 조직들의 사용자들을 지원하기 위하여 설계되었다. 이러한 워크플로우의 구조는 최대 십만 명 정도의 사용자 지원이 가능하나, 수백만 이상의 사용자를 지원해야 하는 인터넷 규모의 서비스 제공에는 성능상의 제약이 따른다.
- 안정성 : 기존의 워크플로우 모델들은 안정적으로 운영되고 있는 서버들에 의하여 수행되고 있음을 가정하고 있다. 그러나 인터넷 규모의 차세대 서비스에서는 다양한 익명의 서비스 제공자들의 협업에 의하여 서비스가 제공됨으로써, 운영 안정성을 보장하기가 불가능하다. 따라서 자주 발생될 수 있는 시스템의 고장에 대비한 복구 기능이 필수적으로 제공되어져야 한다.
- 관리성 : 워크플로우의 필수 기능인 비즈니스 프로세스의 운영 상황을 모니터링하고 통제하는 관리 기능이 제공되어져야 한다.
- 자율성 : 인터넷은 속성상 특정 개인이나 참여자들의 자율적 운영과 관리에 근거하고 있다. 따라서 이러한 자율적인 환경에서 수행되고 운영될 수 있는 워크플로우 모델이 필요하다. 예를 들어 IBM은 이러한 자율성을 강조하여 자율 컴퓨팅(Autonomic Computing)[5]을 주요 전략의 하나로 추진하고 있다.
- 보안성 : 개방적으로 운영되는 인터넷 서비스에서는, 방화벽 내에서 안전하게 운영되어지는 기존의 워크플로우 수행 모델 보다는 더 강화된 보안 체계를 제공해야 한다.

따라서 이 연구에서는 차세대 인터넷 서비스 환경의 주요 요구사항 중 가장 핵심 문제인 확장성과 신뢰성의 제약을 극복할 수 있는 P2P 기반 워크플로우 모델을 연구하고, 보다 진보된 형태의 협업기능 제공을 위한 데이타 구조와 소프트웨어 구조에 대하여 제시하고자 한다.

제2장에서는 워크플로우, 차세대 인터넷 서비스 관련 워크플로우 기술 및 P2P 기술에 대한 관련 연구에 대하여 정리하고, 제3장에서는 P2P 모델 적용의 근거를 논의 한 후, 워크플로우 측면에서 인덱스 정보를 기술한 인스턴스 버디의 개념과 이를 적용한 P2P 기반 워크플로우 모델의 전반적인 구조에 대하여 기술한다. 제4장에서는 불안정한 인터넷 서비스 환경에서 신뢰성 지원을 위한 복구 기능을 수학적으로 정의하고 이에 대한 증명을 다룬다. 제5장에서는 P2P모델에 워크플로우의 기능을 부가하였을 때의 성능 평가를 위하여, 분산 사건 시뮬레이션 도구인 SimPy를 사용한 성능 시뮬레이션과 그 결과에 대하여 정리한다.

2. 관련 연구

관련 연구는 크게 워크플로우 수행 모델을 중심으로 기술하고, 그리드 및 웹 서비스 연구 분야에서 이루어진 워크플로우 연구에 대하여 정리한다. 마지막으로 P2P관련 연구를 구성 방법에 따른 분류를 중심으로 서술한다.

2.1 워크플로우 수행 모델

워크플로우 관리 시스템이 클라이언트/서버 구조 즉 중앙 서버를 기반으로 구현되는 가장 중요한 이유는 모니터링 기능의 구현을 보다 용이하게 하기 위한 것이다. 다수의 중앙 집중식 상용 제품을 개발한 경험을 기반으로 WfMC(Workflow Management Coalition)에서는 최초의 워크플로우 표준인 WfMC 참조 모델[6]을 만들었다. 이 참조 모델에는 하나의 워크플로우 엔진, 주변 구성 요소, 이들 간의 인터페이스 및 용어에 대한 표준안을 제정하였다. 이 당시에는 많은 워크플로우 제품들이 이미 상용화된 상태였으므로, 프로세스 정의 표준인 인터페이스 1을 적용하기가 재개발 비용 문제로 사실상 불가능하였다. 그러나 인터페이스 4는 다른 워크플로우 엔진과의 상호 운영성에 필요한 요소들을 정의하였기 때문에 많은 주목을 받게 되었다. 참조 구현으로써 MIME(Multipurpose Internet Mail Extension)을 적용한 인터페이스 4가 구현되었지만, 전자메일은 신뢰할 수 없는 통신 채널이기 때문에 전송 시 발생되는 예기치 않은 문제에 대한 처리에 많은 어려움이 있었다[7]. 현실적으로 중앙 집중식 접근 방법은 비효율적이며 신뢰성과 확장성에 많은 심각한 문제를 발생시킨다[8].

분산 객체 기술인 CORBA(Common Object Request Broker Architecture) 표준을 제정한 OMG(Object Management Group)에서는 CORBA 환경에 워크플로우의 기능을 추가하기 위한 표준으로 jFlow(Joint Flow)을 제정하였다[9]. 워크플로우 표준의 제정을 위하여 WfMC의 참조 모델을 적용하고 구현 기술을

CORBA/IIOP를 사용함으로써, 이후에 제작된 CORBA 기반의 분산 워크플로우 제품에 많은 영향을 주었다. CORBA가 분산 객체 환경이므로 하나의 논리적인 워크플로우를 구성하면서 동시에 하나의 엔진이 여러 대의 물리적 서버에 분산되는 효과를 기대할 수 있었다. 워크플로우의 구조를 개선하여 자원에 대한 경합을 약화시킴으로써 성능의 향상을 기대할 수 있다[10]. 그러나 제한된 물리적 서버를 다수의 사용자가 공유하는 환경에서는 자원의 경합을 약화시키는데 한계가 있다. 이 논문에서는 기존 분산 워크플로우 모델의 확장성 제한에 대하여 시뮬레이션 단원에서 논의할 것이다.

웹의 확산으로 HTTP 기반의 워크플로우 표준으로 SWAP(Simple Workflow Access Protocol)[11]이 발표되었다. SWAP는 WfMC의 참조 모델을 지원하지 못하는 관찰자/수행자 모델의 단순한 형태로 유지되다가 독자적으로 발전하지 못하였고, 후에 인터페이스 4의 XML 표준인 Wf-XML로 통합 정리되었다[12]. Wf-XML은 워크플로우 시스템간의 상호운용을 위한 XML 기반 프로토콜로서 상호운용을 위한 최소한의 메시지 집합을 제공한다.

이후에, Rusinkiewicz가 휴스턴 대학에서 개발된 INCAs(INformation CArriers)[13]와 같은 에이전트 기반 워크플로우 또는 완전 분산 워크플로우 모델이 제안되었다. 이 모델은 비정형(Ad-hoc) 워크플로우 프로세스를 지원하기 위하여 개발되었다. 이 구조에서 프로세스 정의와 워크플로우 데이타가 INCA라는 컨테이너에 저장되어 에이전트 간에 전송되고, 각 에이전트는 하나의 워크플로우 엔진을 내장하고 작업을 자율적으로 처리한다. 이 시스템에서는 인스턴스를 생성하는 에이전트를 케이스 조정자(case coordinator)라고 하여 그 인스턴스를 관리하는 일종의 서버 역할을 하게 된다. 이 모델은 완전 분산 워크플로우 모델이기 때문에 매우 높은 확장성을 지니고 있으나, 워크플로우의 가장 중요한 기능인 전역적인 모니터링과 관리 기능에 대한 고려가 전혀 없고, 모든 에이전트가 안정되게 운영되고 있다는 가정을 하고 있기 때문에, 차세대 인터넷 서비스에는 실제 적용이 불가능하다.

WONDER 시스템[14]은 INCAs의 모델을 실제로 구현한 것으로 INCAs가 능동적인 에이전트에서 수행되는 수동적인 시스템이라고 한다면, WONDER는 수동적인 에이전트에서 수행되는 능동적 객체를 사용한다. INCAs 모델에서는 감사, 모니터링, 그리고 수행자의 동적 할당 등에 대한 기능 고려가 없다. WONDER가 INCAs의 개념을 구현한 것이지만 관리 기능 구현을 위하여 프로세스 조정자, 역할 조정자, 백업 관리자 등의 특수한 구성 요소들을 지원하고 있다. INCAs와 WONDER는 트랜잭션 문제 발생에 대한 고려만 있을 뿐, 에이전트 즉 시스템 자체의 문제 발생에 대한 고려가 전혀 없다. 즉 케이스 조정자, 프로세스 조정자, 역할 조정자등에 문제가 발생하였을 경우에는 전체 시스템이 마비되는 현상이 발생되며, 이러한 문제가 발생될 확률은 중앙집중식 보다 더욱 커지게 된다. 또한 특수한 구성 요소에 집중되는 부하는 확장성의 문제를 야기한다.

2.2 그리드 및 웹 서비스에서의 워크플로우

WfMC에서는 워크플로우 정의 표준으로 WPDL (Workflow Process Description Language)을 제정하였고, 후에 XML을 적용하여 XPDL(XML Process Description Language)로 재개정되었다. XPDL은 가장 중요한 워크플로우 정의표준으로 인정되었지만, 다수의 표준화 단체들은 적용 분야에 따라 독자적인 프로세스 정의 표준들을 제정하였는데 대표적인 것으로 ebXML의 BPSS(Business Process Specification Schema)와 RosettaNet의 PIP(Partner Interface Process) 등이 있다[15].

웹 서비스의 출현과 급속한 표준화는 UDDI, SOAP (Simple Object Acess Protocol), WSDL(Web Service Description Language)과 같은 기본 기능의 표준만 아니라 워크플로우 기능 제공을 위한 표준으로써 여러 단체 및 회사에서 다양한 웹 서비스를 위한 워크플로우 정의 표준을 제안하였다. HP의 WSCL(Web Service Conversation Language), SUN/BEA의 WSCI (Web Services Choreography Interface), IBM의 WSFL, 그리고 MICROSOFT의 XLANG 등의 웹 서비스 프로세스 정의 모델이 발표되었고, 최근에는 가장 대표적인 웹 서비스용 워크플로우 정의 모델인 WSFL과 XLANG이 BPEL4WS(Business Process Execution Language for Web Services)로 통합되었다[16].

그리드 컴퓨팅 분야에서도 최근에 워크플로우 기능을 제공하기 위한 모델을 연구하고 있다. GSFL은 그리드 서비스를 위한 워크플로우 언어로 웹 서비스를 위한 워크플로우 정의 모델인 WSFL과 그리드 서비스 기술을 기반으로 구축하려고 노력하였다. 확장성을 위하여 P2P 모델 효과에 대하여 강조하였고, 웹 서비스용 워크플로우에서 부족했던 관리 기능에 대하여 언급하였다. 아직 초기 단계이기 때문에 많은 부분이 보완되어야 하며, 향후 표준화에 이르기까지는 많은 경쟁을 거쳐야 하는 상황이다. GridAnt[17]는 자바의 프로그램 구축 도구인 Ant를 기반으로 작성된 클라이언트 측면의 워크플로우 시스템이다. 개념은 아파치의 Ant에 존재하는 워크플로우와 유사한 엔진을 사용하여 그리드 타스크를 개발하는 것이다. 그러나 이 기술은 강력하고 뛰어난 기능을 갖는 기존의 워크플로우 모델을 대체시키기 보다는, 간

단한 그리드 타스크를 테스트하고 개발하는 목적으로 개발되고 있다.

2.3 P2P

P2P에서는 그 구성 방식, 즉 토폴로지에 따라 다음과 같이 서버 기반 P2P, 순수 P2P, 그리고 복합 P2P로 분류된다[18].

- 서버 기반 P2P : 중앙 서버에 전적으로 의지하는 전형적인 P2P 구성 방식으로 가장 많이 사용되는 모델이다. 서버는 검색, 색인 기능, 사용자 인증 및 조정자 역할을 담당한다. 모든 서비스 요청이 서버를 통하여 접근하므로 프로토콜의 구현, 개선, 보안, 그리고 과금 기능을 쉽게 구현할 수 있는 장점이 있는 반면, 서버의 고장이 전체 시스템을 중지시키는 문제가 있으므로 이에 대한 보완이 필요하다.
- 순수 P2P : 중앙 서버가 없는 피어들의 네트워크이다. 중앙 서버가 없으므로 피어들은 자율적으로 인접 피어들과 메시지를 교환함으로써 자원 검색을 수행한다. 특정 노드의 고장이 전체 시스템을 중지시키지 않는 장점이 있으나, 검색을 위한 과다한 메시지 전송으로 인한 네트워크 부하 문제가 있으며 프로토콜의 갱신이 어렵다. 가장 큰 문제 중 하나는 피어가 시스템에 연결될 때, 피어 네트워크에 접속하기 위하여 최소 하나의 연결된 피어를 알고 있어야 한다는 점이다. 대표적인 예로서 누텔라[19]와 프리넷[20]이 있다.
- 복합 P2P : 연결되어진 피어들 중에서 선택된 피어를 서버(슈퍼 노드)로 사용하는 구성 방식을 갖는다. FastTrack[21]에서 최초로 복합 P2P 방식을 사용하였다. 피어가 슈퍼 노드에 연결될 때, 모든 파일의 리스트가 슈퍼 노드에 저장하고, 피어는 자신의 슈퍼 노드에게만 검색 요청을 수행하며, 슈퍼 노드는 다른 슈퍼노드들에게 파일 검색 요청을 재 전송한다. 슈퍼 노드들 간의 연결은 순수 P2P 모델의 방법을 사용한다. 이러한 구성 방식은 서버 기반 구성 방식의 효율성과 순수 P2P 방식의 안정성을 제공받을 수 있으나, 슈퍼 노드의 자율적이고 효율적인 구성을 위하여 많은 연구가 필요하다.

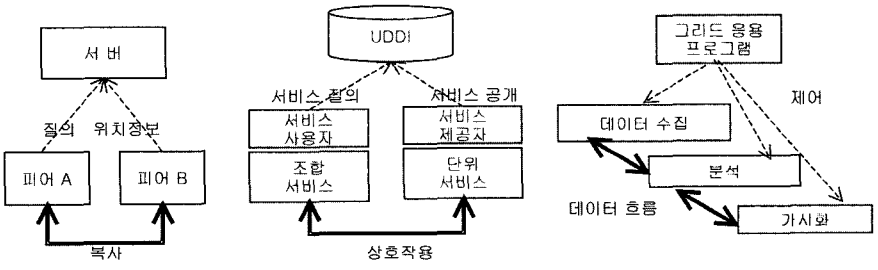
3. P2P 기반 워크플로우 모델

제3장에서는 P2P 모델과 차세대 인터넷 서비스의 유사성에 대하여 살펴보고, P2P 기반 워크플로우의 기본적인 구조와 원리에 대하여 기술하고, P2P 워크플로우 시스템에 있어서의 가장 기본이 되는 인덱스 정보인 인스턴스 버디를 정의하고, 이를 예제를 통하여 설명한다.

3.1 차세대 인터넷 서비스와 P2P 모델과의 유사성

이 연구에서는 인터넷 규모의 다수의 사용자를 지원하는 실행 모델을 구축하기 위하여, P2P 모델을 고려하였다. P2P 기술을 고려한 이유는 차세대 인터넷 서비스 모델이 P2P 모델과 높은 유사성을 가질 뿐만 아니라, 이 기술이 인터넷에서 협업 기능을 제공하는 모델 중에서 거의 유일하게 실제적으로 인터넷 규모의 참여자를 지원하고 있기 때문이다.

그림 1에서와 같이, P2P 기술은 차세대 인터넷 서비스들과 매우 유사한 서비스 구조를 갖고 있다. 그림 1의 (가)는 일반적인 P2P 구조를 보여주는데, 서버를 통해서 자원이 있는 피어를 찾아낸 후 자원을 보유하고 있는 피어에서 직접 자원을 복사한다. 그림 1의 (나)는 웹 서비스 구조를 보여준다. 웹 서비스도 UDDI(Universal Description, Discovery, and Integration)에서 서비스의 위치를 찾아내고, 서비스 제공자와 직접적인 상호작용을 통하여 서비스를 제공받는다. 프로세스간의 협력은 서비스 사용자와 제공자 사이에 또는 서비스 제공자들 사이의 직접적인 상호 작용으로 이루어진다. 그림 1의 (다)에서는 그리드 컴퓨팅의 예를 보여주고 있는데, 특히 대량의 데이타를 처리해야 하기 때문에 그리드 응용 프로그램은 서버 또는 조정자 역할을 하며 실제 데이타의 전송이나 처리는 컴퓨팅 노드들 사이에서 이루어져야한다. 이러한 유사성으로 인하여 그리드 컴퓨팅과 웹 서비스는 P2P 모델을 기본으로 하고 있음을 알 수 있



(가) P2P (나) 웹 서비스 (다) 그리드 컴퓨팅

그림 1 차세대 인터넷 서비스와 P2P의 유사성

다. 이에 따라 IBM에서는 최근에 웹 서비스와 그리드 컴퓨팅의 통합을 추진 중이다.

이들의 공통점을 살펴보면, 서버는 인덱스 정보만을 저장하고 전체적인 조정 역할만 수행하며, 실제 데이타의 처리나 전송은 피어들 간에 수행되어짐으로써 서버의 부하를 최소화시킨다. P2P 기술은 클라이언트의 자원의 활용을 극대화하고 서버의 역할을 최소화함으로써 인터넷 규모의 확장성을 지원할 수 있다. 그러나 클라이언트인 서비스 제공자 및 참여자들이 데스크 탑 컴퓨터나 노트북 컴퓨터와 같이 안정적이지 못하다는 문제가 있으며, 제공되는 서비스의 수준이 파일 공유 정도의 단순한 상태라는 문제점을 갖고 있다. 이러한 문제점에도 불구하고 모델의 유사성을 볼 때 P2P 기반의 워크플로우 기술은 웹 서비스나 그리드 컴퓨팅에 있어서 필수불가결한 기술이라고 판단할 수 있다.

3.2 피어플로우의 개요

워크플로우에 적용될 P2P 모델을 살펴보면, 순수 P2P 모델은 관리 기능의 구현이 어려우며, 과도한 자원 탐색 메시지 처리 문제로 확장성이 매우 부족하다. 복합 P2P 모델은 상용제품인 FastTrack에서 처음 적용된 모델로, 기술의 보안을 이유로 자세한 구조나 특성이 아직 공개되어 있지 않은 상황이다. 따라서 이 연구에서는 구현 및 서버 관리의 용이성이 높은 서버 기반 P2P 모델을 우선 적용하였다.

피어플로우는 그림 2에서와 같이 하나의 인덱스 서버와 다수의 피어들로 구성되어 있다. 피어플로우에서 인덱스 서버는 프로세스의 정의 시 기존 워크플로우 서버와 동일한 역할을 수행하며, 프로세스의 수행 시에는 각 프로세스 인스턴스의 실제 데이타가 아닌 인덱스 정보만을 관리함으로써 일종의 조정자 역할을 수행한다. 이러한 방식은 P2P 방식을 사용하는 모든 시스템들이 공통적으로 갖는 특성이다. 각 피어들은 자율적인 하나의 워크플로우 시스템으로 작업 처리와 라우팅(작업 흐름 결정)을 서버의 도움 없이 수행하게 된다. 기존의 분산 워크플로우 시스템에서는 서버를 구성하고 있는 요소들이 고유한 자신의 역할을 갖고 있지만, P2P 기반 워크플로우에서 인덱스 서버를 제외한 모든 피어는 동일한 역할을 수행함으로써 자율적인 관리 기능을 보다 쉽게 구현 할 수 있다.

그림 2에서 굵은 실선으로 표시된 화살표는 대부분의 실제 정보 교환이 피어들 간에 발생하여 서버의 성능에 영향을 주지 않음을 의미하며, 점선은 전역적 조정을 위하여 피어들과 인덱스 서버 간의 소량의 정보 교환이 이루어짐을 나타낸다. WfMC 참조모델에서는 모든 정보교환이 중앙의 서버를 통하여 이루어지기 때문에 확장성이 서버의 성능에 종속되어지는 문제점이 있는 반면, 이 구조는 서버의 부하를 최소화시키고 피어의 사용을 극대화시킴으로써 확장성을 높일 수 있는 구조를 제공한다.

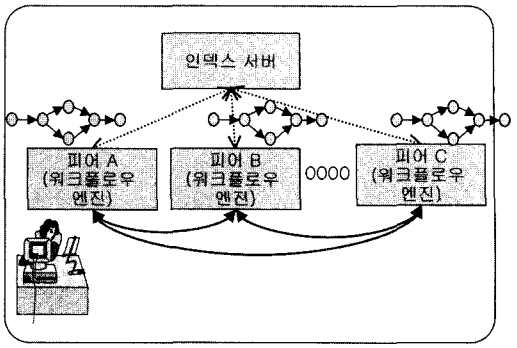


그림 2 피어플로우의 개념도

기존의 워크플로우 모델의 경우, 서버와 사용자가 1:N 관계를 갖는 것과는 달리, 피어플로우에서는 각 사용자의 시스템이 워크플로우 엔진을 내장하고 있기 때문에 1:1 관계를 갖는다. 일반적으로 워크플로우 서버는 다수의 사용자들에 대한 연결 관리를 처리해야 하고, 모든 사용자의 데이타를 관리해야 하는 비 기능적 요구사항을 만족시켜야함으로 복잡성이 크게 증가된다. 따라서 하나의 사용자와 그 사용자의 데이타를 처리하기 위한 워크플로우의 규모는 일반적인 피어들의 자원으로도 충분히 지원이 가능할 정도로 작게 구현될 수 있다. INCAs, WONDER, 그리고 GridAnt에서도 워크플로우 엔진의 역할이 하나의 작은 에이전트 내에서 구현될 수 있음을 보여주고 있다. 구성 요소들 간의 데이타 전송은 JXTA, SOAP 그리고 Globus와 같은 응용 분야에 적합한 프로토콜이 사용될 수 있으나, 이 논문에서는 프로토콜에 대한 부분보다는 모델에 중점을 두고 기술하고자 한다.

3.3 피어플로우의 구조

그림 3은 WfMC의 워크플로우 제품의 일반적 구조를 기반으로, 기존 서버의 역할이 인덱스 서버와 피어들 간에 어떻게 할당 되는가를 나타내기 위하여, 제안하는 시스템의 구조를 보여준다. 서버의 역할을 인덱스 서버와 피어로서 재배치를 하고, 그들 간의 통신을 위하여 영속적 메시지 큐를 제공하였다. 메시지 큐를 사용함으로써 연결 관리를 용이하게 제공할 수 있다.

일반적으로 P2P 시스템은 실제 데이타와 인덱스 정보를 명확히 구분한다. 예를 들어, Napster에서는 파일의 이름이 인덱스 정보이고, 파일 자체가 실제 데이타가 된다. 피어플로우에서의 인덱스 정보는 하나의 워크플로우, 즉 프로세스 인스턴스에 참여하고 있는 피어들의 식별자의 집합인 인스턴스 버디이며 실제 데이타는 워크

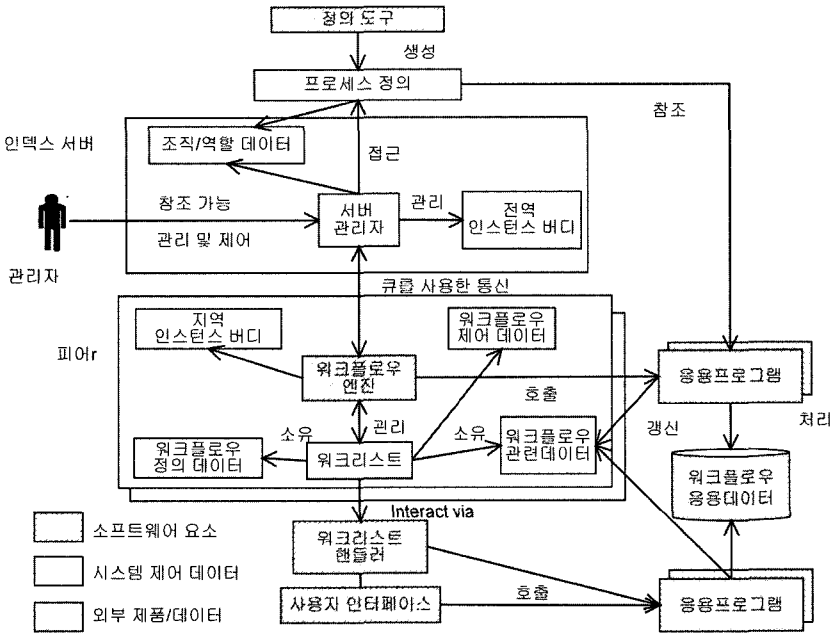


그림 3 피어플로우의 구조

아이템이 된다.

피어 플로우에서는 이러한 인스턴스 버디를 전역 인스턴스 버디와 지역 인스턴스 버디에 저장한다. 서버에 저장되는 전역 인스턴스 버디는 모든 인스턴스 버디를 포함하는 집합으로 시스템 관리자나 업무 관리자에게 전역적 모니터링 기능을 제공하거나 시스템의 복구 시 사용된다. 관련 피어들 간에 저장되는 지역 인스턴스 버디는 피어들이 서버의 도움 없이 자율적으로 업무를 처리하기 위한 인덱스 정보의 역할을 갖는다.

실제 워크플로우 엔진에서 처리하는 데이타인 워크아이템은 서버에는 저장되지 않으며, 각 피어들의 워크리스트 내에 저장된다. 기존의 WfMC 참조 모델에서는 워크아이템이 저장하는 데이타로 인스턴스의 내부 상태 정보인 워크플로우 제어 데이타(Workflow Control Data)와 라우팅 조건 검사 시 사용하는 워크플로우 관련 데이타(Workflow Relevant Data)가 있다. 그러나 피어플로우에서는 자체적인 라우팅 처리를 위하여 IN-CAs와 같이 워크플로우 정의 데이타를 추가 하였다. 이를 통하여 각 피어들은 서버의 도움 없이 워크플로우의 라우팅을 스스로 처리할 수 있으며, 워크아이템의 변경 시에 지역 인스턴스 버디를 참조하여 데이타의 변경사항을 관련된 피어들 간에 동기화시킬 수 있다.

그림 4는 인스턴스 버디들에 대한 예를 보여주고 있다. 이 시스템 내의 워크플로우 인스턴스로 I_1, I_2, I_3 3개가 있으며, 인스턴스 I_1은 피어 A와 피어 B가 참여하고 있으며, I_2와 I_3는 피어 B와 피어 C가 참여하고 있는 것을 가정한다. 따라서 전역 인스턴스 버디에는 I_1, I_2와 I_3의 인스턴스 버디들이 모두 저장되어 있다. 예를

인덱스 서버		
전역 인스턴스 버디	I_1 I_2 I_3	P_A, P_B P_B, P_C P_B, P_C

피어 B		
지역 인스턴스 버디	I_1 I_2 I_3	P_A, P_B P_B, P_C P_B, P_C
워크리스트	d_1, d_2, d_3	

피어 A		
지역 인스턴스 버디	I_1	P_A, P_B
워크리스트	d_1	

피어 C		
지역 인스턴스 버디	I_2 I_3	P_B, P_C P_B, P_C
워크리스트	d_2, d_3	

그림 4 인스턴스 버디의 예

들어, 피어 C에는 I_2와 I_3에 참여하고 있으므로 지역 인스턴스 버디에 I_2와 I_3의 인스턴스 버디들이 저장되어 있다. 또한 피어 C의 워크리스트에는 지역 인스턴스 버디에 관련는 워크아이템인 d_2, d_3이 저장된다. 이로서 인스턴스 버디들은 인덱스 서버내의 전역 인스턴스 버디와 각 피어들의 지역 인스턴스 버디들 간에 중복 저장됨을 알 수 있고, 또한 인스턴스 버디들과 워크아이템들은 피어들 간에 중복 저장되고 있음을 알 수 있다. 예를 들어 I_1의 인스턴스 버디는 전역 인스턴스 버디와 피어 A, 피어 B에 중복 저장되어 있으며, I_1의 인스턴스 버디와 워크아이템 d_1은 피어 A와 피어 B 간에 중복된다.

4. 피어플로우의 시스템 복구

인터넷 규모의 컴퓨팅 환경에서는 일반적인 워크플로우 환경보다 시스템의 안정성 측면에서 많은 문제가 발생될 가능성이 있기 때문에, 시스템의 복구 기능은 필수적인 기능이다. 제4장에서는 제3장에서 설명한 피어플로우의 구조와 인스턴스 버디를 통하여 피어플로우가 제공하는 복구 기능을 기술함으로써, 피어플로우의 안정성 측면을 서술한다.

피어플로우에서는 인스턴스 버디와 워크아이템이 인덱스 서버와 피어들 사이에 중복되어 있기 때문에, 피어 또는 인덱스 서버에 고장이 발생되는 경우, 중복 데이타를 이용하여 복구 기능을 제공해 줄 수 있다. 지금까지 워크플로우 연구에 있어서 트랜잭션 단위의 문제 발생에 대한 복구 기능에 대해서는 많은 연구가 있어왔지만, 기존 모델에서는 데이타의 중복을 모델에서 고려하지 않았기 때문에 시스템 단위의 복구 기능 제공에 대한 적극적인 연구는 미진하였다. 복구기능에 대한 수학적 증명을 위하여, 먼저 주요 용어에 대하여 다음과 같이 정의하였다.

정의 1. 전역 인스턴스 버디n개의 피어들의 식별자의 집합을 $P = \{p_1, p_2, ..., p_n\}$, m개의 인스턴스들의 식별자의 집합을 $I = \{i_1, i_2, ..., i_m\}$, 그리고 각 인스턴스들의 실제 데이타인 워크아이템의 집합을 $D = \{d_1, d_2, ..., d_m\}$라고 하였을 때. 전역 인스턴스 버디 G를 다음 같이 정의한다.

$$G \subseteq P \times I$$

정의 2. 피어 i의 지역 인스턴스 버디전역 인스턴스 버디와 G를 상기와 같이 정의하였을 때, 피어 i의 지역 인스턴스 버디 L_i를 다음 같이 정의한다.

$$L_i = \{(x, y) \in G \mid x = p_i \vee (p_i, y) \in G\}$$

정의 3. 피어 i의 워크리스트전역 인스턴스 버디 G와 피어 i의 지역 인스턴스 버디 L_i를 상기와 같이 정의하였을 때, 피어 i의 워크리스트 W_i를 다음과 같이 정의한다.

$$W_i = \{ (y, d_i) \in I \times D \mid \exists \ x \text{ such that } (x, y) \in L_i\}$$

예를 들어, $P = \{A, B, C, D\}$, $I = \{1, 2, 3\}$, $D = \{d_1, d_2, d_3\}$, 그리고 $G = \{(A, 1), (B, 1), (B, 2), (C, 2), (C, 3), (D, 2)\}$라고 가정하였을 때, 각 피어의 지역 인스턴스 버디를 살펴보자. $A = p_1$이므로 L_i의 첫 번째 조건을 만족하므로 $(A,1) \in G$, $(A,1) \in G$이므로 L_i의 두 번째 조건을 만족하므로 $(B,1) \in L_1$가 된다. 반면 $(B, 2)$는 두 가지 조건을 모두 만족하지 않으므로 $(B, 2) \notin L_1$가 된다. 이와 같은 방법으로 $L_1 = \{(A, 1), (B, 1)\}$이 된다. 같은 방법으로 $L_2 = \{(A, 1), (B, 1), (B, 2), (C, 2), (D, 2)\}$, $L_3 = \{(B, 2), (C, 2), (D, 2), (C, 3)\}$, 그리고 $L_4 = \{(B, 2), (C, 2), (D, 2)\}$가 됨을 알 수 있다. 각 피어의 워크리스트를 구해보면 $W_1 = \{(1, d_1)\}$, $W_2 = \{(1, d_1), (2, d_2)\}$, $W_3 = \{(2, d_2), (3, d_3)\}$, 그리고 $W_4 = \{(2, d_2)\}$임을 알 수 있다.

4.1 인덱스 서버의 복구

인덱스 서버 시스템에 문제 발생시 복구에 대하여 살펴보면, 인덱스 서버에서 가장 동적이며 중요한 데이타는 전역 인덱스 버디이다. 시스템 복구의 경우 다른 데이타들은 정적이고 쉽게 백업되고 복구 될 수 있으므로 이 연구에서는 전역 인덱스 버디들을 복구함으로써 인덱스 서버가 복구 될 수 있다고 가정하였다. 다음 정리는 전역 인스턴스 버디들이 지역 인스턴스 버디들로부터 재생성될 수 있음을 보여준다.

정리 1. 전역 인스턴스 버디의 복구

$P = \{p_1, p_2, ..., p_n\}$, $I = \{i_1, i_2, ..., i_m\}$이고 $L_i = \{(x, y) \in G \mid x = p_i \vee (p_i, y) \in G\}$인 경우 지역 인스턴스 버디의 집합족을 $L = \{L_1, L_2, ..., L_n\}$이라고 가정하면, 전역 인스턴스 버디 $G = \bigcup_{i=1}^{n} L_i$

증명. L_i의 정의에 따르면 $L_i = \{(x, y) \in G \mid x = p_i \vee (p_i, y) \in G\}$이다. 따라서 각 L_i는 G에 포함된다. 따라서 $\bigcup_{i=1}^{n} L_i$도 역시 G에 포함된다. 반대 방향을 증명해 보자. 공집합은 모든 집합의 부분집합이므로 G가 공집합이 아님을 가정할 수 있다. 따라서 G의 원소가 되는 (x, y)를 선택할 수 있다. $x \in P$이므로 $x = p_j$인 j가 존재하며 이는 L_j 정의의 첫 번째 조건을 만족시킴을 알 수 있다. 따라서 $(x, y) \in L_j$이고 $L_j \subseteq \bigcup_{i=1}^{n} L_i$이므로 $(x, y) \in \bigcup_{i=1}^{n} L_i$가 된다.

4.2 피어들의 복구 조건

피어에 있어서 복구되어야할 가장 중요한 데이타는 지역 인스턴스 버디와 각 인스턴스에 대한 실제 데이타인 워크아이템이다. 전역 인스턴스 버디를 갖고 있으면

지역 인스턴스 버디의 정의인 $L_i = \{(x, y) \in G \mid x = p_i \vee (p_i, y) \in G\}$에 따라서 지역 인스턴스 버디는 쉽게 복구할 수 있다. 또한 지역 인스턴스 버디를 갖고 있으면, 실제 워크아이템을 갖고 있는 피어들을 알 수 있기 때문에 워크아이템도 쉽게 복구할 수 있다.

그러나 하나 이상의 피어들에 이상이 발생되었을 때는 복구할 수 없는 경우도 있다. 예를 들어 인스턴스 i_3에 관여하고 있는 피어들로 p_1, p_4가 있다고 가정하자. 만약 p_1에 고장이 발생한 경우에는 p_4로부터 i_3의 워크아이템을 복구할 수 있지만, p_1, p_4 두개 모두 고장이 발생한 경우에는 중복 저장된 모든 워크아이템이 파괴되었으므로 때문에 복구가 불가능하다. 따라서 다음 정리에서는 어떤 피어들이 고장이 발생했을 때 복구할 수 있는가에 대한 조건을 보여준다. 피어 i의 워크리스트의 정의 $W_i = \{(y, d_i) \in I \times D \mid \exists x \text{ such that } (x, y) \in L_i\}$에 따라 특정 피어가 관여하고 있는 인스턴스 y에 대한 워크아이템이 항상 피어 i의 워크리스트에 존재하게 된다. 따라서 아래 정리에서는 워크아이템의 복구 조건을 지역 인스턴스 버디의 복구 조건으로 증명하였다.

정리 2. 피어의 복구 조건

L'을 고장이 발생한 피어들의 지역 인스턴스 버디들의 집합족이라고 하고, L''을 고장이 발생하지 않은 피어들의 지역 인스턴스 버디들의 집합족이라고 하자. 즉 $|L'| = n$, $|L''| = m$, $n+m = k = |L|$, $L' \cup L'' = L$, $L' \cap L'' = \varnothing$이라고 가정할 때, 만약 $\bigcup_{i=1}^{n} L' \subseteq \bigcup_{i=1}^{m} L''$이면 $\bigcup_{i=1}^{m} L'' = G$이다.

증명. $L' \cup L'' = L$이므로 $L'' \subseteq L$이다.

따라서 $\bigcup_{i=1}^{m} L'' \subseteq \bigcup_{i=1}^{k} L = G$

반대방향을 증명해보자. 공집합은 모든 집합의 부분집합이므로 G가 공집합이 아님을 가정할 수 있다. $(x, y) \in G$인 (x, y)를 선택할 수 있으며, $(x, y) \in \bigcup_{i=1}^{n} L'$이거나 $(x, y) \in \bigcup_{i=1}^{m} L''$이다. 만약 $(x, y) \in \bigcup_{i=1}^{n} L'$이면 $\bigcup_{i=1}^{n} L' \subseteq \bigcup_{i=1}^{m} L''$이므로 $(x, y) \in \bigcup_{i=1}^{m} L''$이다. 따라서 모든 $(x, y) \in G$에 대하여 $(x, y) \in \bigcup_{i=1}^{m} L''$이다. 이것은 $G \subseteq \bigcup_{i=1}^{m} L''$임을 의미한다.

4.3 인덱스 서버와 피어에 이상이 동시에 발생하는 경우

마지막으로 피어와 인덱스 서버가 동시에 이상이 발생되는 경우를 생각해 볼 수 있다. 이 경우 인덱스 서버는 이상이 발생된 피어들이 완전하게 복구된다면 정리 1에 의하여 복구가 가능하다. 이상이 발생된 피어들의 완전 복구 여부는 정리 2에 따르게 된다. 또한 완전한 복구가 아니더라도 이상이 발생된 피어 P_j는 부분적으로 복구될 수 있다. 즉 $(x, y) \in L_j$인 (x, y)에 대하여 $(x, y) \in \bigcup_{k=1}^{m} L''$인 경우 인스턴스 i_y는 복구가 가능하다.

지금까지 워크플로우의 복구에 대한 연구는 INCAs나 WONDER에서도 하나의 트랜잭션의 복구 위주로 연구되었다. 피어플로우에서는 시스템 자체에 결함이 발생 때에도 적용될 수 있음을 알 수 있다. 이러한 기능은 특히 결함이 자주 발생되는 인터넷 환경에서 매우 유용하게 활용될 수 있는 기능이며 필요시 인덱스 서버가 동적으로 대체될 때도 사용될 수 있다.

5. 피어플로우의 성능 평가

일반적으로 P2P 모델은 전통적인 클라이언트/서버 모델보다 약 1000배정도의 성능 향상을 기대할 수 있다고 한다. 제5장에서는 P2P모델을 워크플로우에 적용하였을 때도 이러한 성능 향상의 가능한가에 대하여 다음과 같은 환경의 시뮬레이션으로 확인해 보았다.

5.1 시뮬레이션 환경

5.1.1 시뮬레이션 환경 및 도구

이벤트 시뮬레이션 모델(Discrete Event Simulation Model)을 제공하는 SimPy[22]를 사용하였다. 이것은 Simula 언어의 개념을 발전시킨 시뮬레이션 패키지다. SimPy를 선택한 이유는 Python으로 만들어진 유일한 시뮬레이션 패키지로 무료로 사용이 가능하며, 이산 사건 시뮬레이션을 완전하게 지원하는 몇 안 되는 객체지향 시뮬레이션 도구 중 하나이기 때문이다. 하드웨어는 Intel Pentium 4 2.4GHz 시스템을, 운영체계는 MS Windows 2000 Server를 사용하였다.

5.1.2 시뮬레이션 모델

기존 모델과의 비교를 위하여 중앙집중식 워크플로우 모델과 분산 워크플로우 모델을 P2P 기반 모델과 같이 비교 하였다. 그림 5는 사용된 시뮬레이션 모델을 보다 자세하게 보여준다. 그림 5의 (가)는 중앙집중식 모델로서 한대의 워크플로우 서버가 데이타베이스 서버를 포함하고 있다. 그림 5의 (나)는 별도의 데이타베이스 서버와 2대의 워크플로우 서버로 구성되어져 있다. 워크플로우 서버의 숫자를 증가시켜가면서 응답시간의 추이를 측정하였다. 그림 5의 (다)는 데이타베이스 서버를 포함

하는 한대의 인덱스 서버와 피어들로 구성되어져 있다. 그림에서의 굵은 선은 자원에 대한 경합이 발생하는 부분을 나타낸다. 작업 생성기(Workcase Generator)는 지수분포 간격으로 시뮬레이션에 필요한 작업을 생성시키는 모듈이다. 워크리스트 처리기는 각 클라이언트에 워크플로우 API를 제공하기 위한 일종의 서버에 대한 에이전트이다. 각 서버는 FCFS의 큐 처리 방식을 갖는 자원으로 구현하였고, 서버의 응답시간을 위주로 측정하기 위하여 네트워크는 서비스 사용자의 숫자에 상관없이, 상수 지연 시간을 갖는 단순 지연 서버로 구현하였다. 참고로 데이타베이스 서버의 서비스 당 수행시간은 2ms, 워크플로우 서버의 서비스 당 수행시간은 5ms로 가정하였다.

5.1.3 측정 기준

대표적인 성능 측정 기준으로는 활용률, 응답시간, 처리율이 있다. 부하를 계속 증가시켜 가면서 처리하였기 때문에 서버의 활용률은 거의 100%에 가까워지게 되므로 제외하였고, 처리율은 응답시간으로 어느 정도 예측 가능함으로 실제 사용자의 입장에서 느끼는 성능인 응답 시간을 측정 기준으로 하였다.

5.1.4 시뮬레이션 시나리오

보다 실제적인 응답시간을 비교하기 위하여 단순히 하나의 프로세스 정의에 대한 시뮬레이션 보다는 실제 서버에 영향을 주는 표 1에서와 같은 서비스들과 이들이 실제 발생할 확률을 작업자가 작업을 하는 다음과 같은 시나리오를 중심으로 기술하였다.

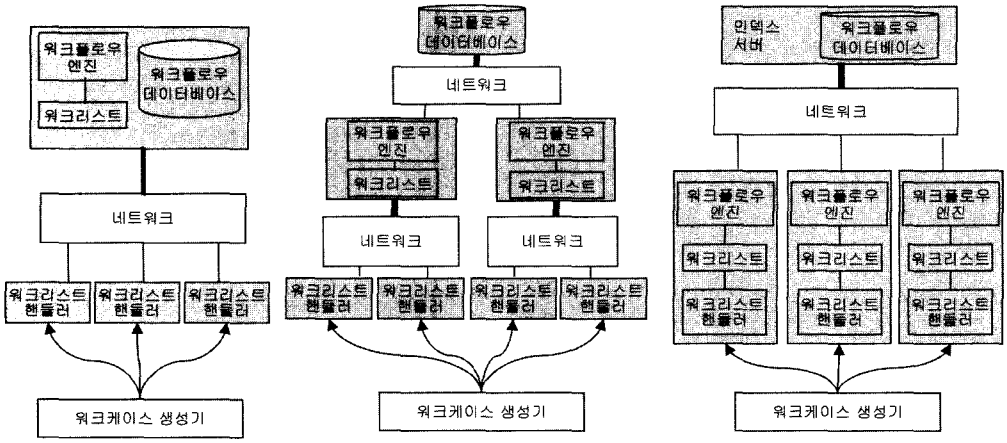
- 작업을 위한 세션 설정
- 프로세스 생성을 위하여 정의 데이타를 전송 받음
- 4개의 관련 피어를 갖는 1개의 인스턴스 생성
- 4번의 작업 처리 전의 인스턴스 정보 확인
- 4번의 각 작업의 처리
- 세션 종료

이 시나리오에 나타나는 20개의 서비스들을 하나의 그룹으로 묶어, 지수분포의 발생 간격으로 단위 시간당 4개의 작업 그룹을 작업 생성기에서 발생시킨다.

표 1 서비스 및 발생 확률

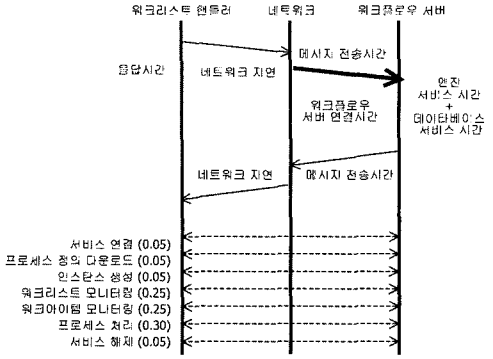
서비스	발생확률
서비스 연결	0.05
프로세스 정의 다운로드	0.05
인스탄스 생성	0.05
워크리스트 모니터링	0.25
워크아이템 모니터링	0.25
프로세스 처리	0.30
연결 해제	0.05

그림 6은 세 가지 워크플로우 모델의 작업 처리에 대한 시나리오를 보여준다. 상단 부분은 처리가 가장 긴 서비스의 순차 다이어그램을 나타내며, 하단 부분은 각 서비스별 발생확률과 수행 범위를 보여준다. 그림 6의 (가)는 중앙집중식 모델에서는 워크플로우 서버가 병목지점이 되며, 모든 서비스가 워크플로우 서버 자원을 요구함을 보여주고 있다. 그림 6의 (나)에서는 워크플로우 서버와 데이타베이스 서버 두개의 병목지점이 존재하며, 중앙집중식 모델과 같이 모든 서비스가 데이타베이스 서버의 자원을 요구함을 볼 수 있다. 그림 6의 (다)에서 보여주는 P2P 기반 모델에서는 인덱스 서버가 하나의 병목지점으로 존재하지만, 발생 확률이 낮은 단지 2개의 서비스가 인덱스 서버의 자원을 요구함을 볼 수 있다.

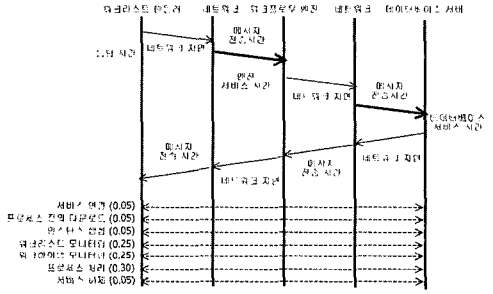


(가) 중앙 집중식 모델 (나) 분산 모델 (다) P2P 모델

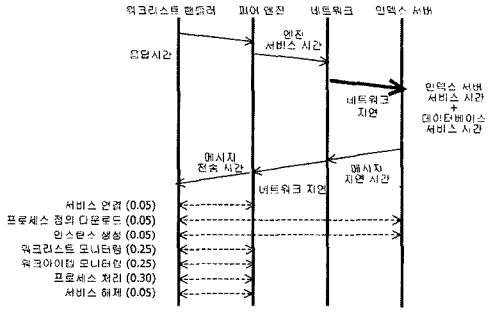
그림 5 성능 시뮬레이션 모델



(가) 중앙집중식 워크플로우 모델



(나) 분산 워크플로우 모델



(다) P2P 기반 워크플로우 모델

그림 6 성능 측정을 위한 시뮬레이션 시나리오

5.2 시뮬레이션 분석 결과

표 2는 각 3가지 모델에 대한 시뮬레이션 결과로서 작업의 수가 증가됨에 따라 발생되는 응답시간의 추이를 보여준다. 그림 7은 표 3을 그래프로 나타낸 것이다. 그림 7에서 첫 번째 라인은 중앙집중식 모델의 응답시간이 작업의 개수가 증가함에 따라 선형적으로 증가함을 보여주고 있다. 두 번째 라인은 하나의 데이타베이스 서버와 2개의 워크플로우 서버로 구성된 분산 워크플로우 모델에 대한 응답시간의 추이를 보여준다. 역시 선형적으로 증가하고 있으나 중앙집중식에 비하여 약 3배의 응답시간 향상을 있음을 알 수 있다. 세 번째 라인은 3개의 워크플로우 서버를 갖는 모델의 응답시간으로써 두 번째 라인보다 약간의 향상이 있음을 볼 수 있다. 네 번째와 다섯 번째 라인의 경우는 각각 4개와, 5개의 워크플로우 서버로 구성되었지만 응답시간의 향상이 없음을 보여준다. 즉, 데이타베이스 서버가 병목지점이 되어 워크플로우 서버의 숫자를 증가시키더라도, 응답시간의 향상에는 한계가 있음을 보여준다. 마지막 라인은 P2P 모델의 응답시간을 보여준다. 실제 서비스의 10%만이 인덱스 서버의 자원을 요구하기 때문에, 같은 하드웨어

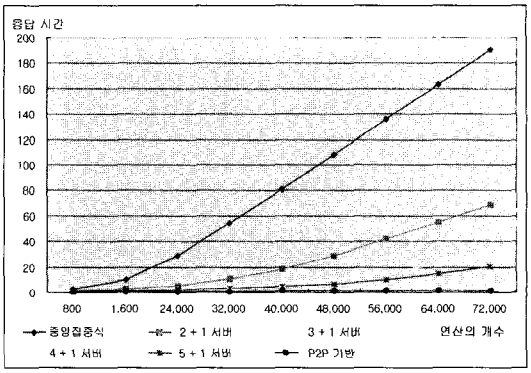


그림 7 모델별 응답 시간 결과 그래프

표 2 모델별 응답 시간 결과 표

서비스 수	중앙 집중식	분산식				P2P 기반
		2+1 서버	3+1 서버	4+1 서버	5+1 서버	
8,000	2.57	1.29	0.96	0.87	0.87	0.72
16,000	9.94	2.73	1.70	1.47	1.47	0.79
240,000	29.46	5.57	2.84	2.31	2.31	0.87
320,000	54.67	10.71	4.36	3.37	3.37	0.95
400,000	81.14	18.24	6.76	4.87	4.87	1.03
480,000	108.05	29.16	10.14	6.89	6.89	1.11
560,000	135.78	42.08	15.65	10.44	10.44	1.19
640,000	163.06	55.18	22.17	14.95	14.95	1.27
720,000	190.31	68.52	29.95	20.31	20.31	1.36
성능 비	1.00	2.78	6.35	9.37	9.37	140.05

를 사용하는 중앙집중식 보다 약 140배의 응답 시간 향상을 제공한다. 만약 자주 사용되는 프로세스의 정의를 피어의 캐쉬에 저장한다면 280배의 성능 향상을 더 기대할 수도 있다. 물론 이러한 시뮬레이션 자료가 정확한 수치를 보여주는 것은 아니지만, P2P 기반의 워크플로우 확장성을 충분히 예측해 볼 수 있다.

6. 결 론

최근 인터넷 환경에서 수행되는 주요 차세대 서비스들을 지원하기 위해서는 기존의 워크플로우 모델은 확장성과 안정성에 매우 큰 제약 사항을 갖고 있다. 이 연구에서는 이러한 제약 사항을 해결하기 위하여 P2P 기반의 워크플로우 모델을 제안하였다. 단순한 P2P 모델에 워크플로우 기능을 제공하기 위하여, 구성 요소들 간의 상호작용을 지원하기 위한 인덱스 정보로서 인스턴스 버디의 개념을 도입하였으며, 워크플로우의 표준인 WfMC 모델을 기반으로 하여 설계하였다.

차세대 인터넷 서비스의 규모를 지원하기 위해서는 기존의 방식과는 차별화된 새로운 접근 방법이 필요하다. P2P 기반의 워크플로우 모델은 피어 내부의 자율적인 처리를 가능하게 하여 서버의 부담을 최소화시켜 확장성을 극대화한다. 이 논문에서는 시뮬레이션을 통하여 기존 모델의 확장성에 대한 제한점을 보여주고 P2P 기반 워크플로우 모델의 성능 향상을 검증하였다. 같은 규모의 자원으로 약 140배의 확장성이 있음을 알 수 있었다.

기존의 워크플로우 모델에서는 단지 트랜잭션 수준의 복구 기능만 제공을 하였으므로, 자주 발생되는 시스템 고장에 대한 복구 기능이 제공되어져야 한다. 이 연구에서는 안정성을 보장할 수 없는 차세대 인터넷 서비스 환경 하에서 수시로 발생할 수 있는 고장에 대하여 인덱스 서버와 피어의 복구가 가능함을 수학적으로 증명하였으며, 다수의 피어들에 고장이 발생하였을 때는 복구할 수 있는 조건을 제시하고 수학적으로 증명하였다. 이를 이용하여 보다 안정적인 서비스를 제공할 수 있을 뿐만 아니라 향후 인덱스 서버를 동적으로 구성하는데도 활용할 수 있다.

워크플로우 시스템의 도입 목적 중 가장 중요한 것이 프로세스 라이프사이클에 대한 관리이다. 이 연구에서 자세하게 언급하지는 않았지만, 인덱스 정보와 워크아이템의 제공을 통하여 워크플로우 관리에 필요한 최소한의 데이타를 지원한다. 현재 여러 가지 차세대 인터넷 서비스들이 발표되고 있으며, 이들 간의 통합도 활발하게 일어나고 있는 상황에서 이러한 서비스들이 현실화될 경우 이 연구에서 제시하는 모델이 매우 유용하게 활용될 것이다.

이 모델은 특정 인터넷 서비스에 독립적인 수행 모델이므로, 향후 표준의 진행 결과에 따른 구현 환경에 대한 보완이 필요하다. 이 연구에서는 서버 중심 P2P 모델을 워크플로우에 적용하였다. 비록 기존의 방식에 비하여 현저한 성능 향상을 보이고 있으나, 아직까지도 특정 서버의 존재는 완벽한 확장성과 자율적 관리를 지원하기에는 부족한 면이 있다. 따라서 피어 자체가 서버의 역할을 할 수 있는 복합 P2P 모델의 적용은 향후 중요한 연구 과제가 될 것이다. 그럼에도 불구하고 인스턴스 버디의 개념 및 복구 개념은 매우 중요하게 활용될 것으로 기대된다.

참 고 문 헌

[1] E. Ceram, "Web Services Essentials," O'Reilly, First Edition February 2002.

[2] I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," International J. Supercomputer Applications, 15(3), 2001.

[3] F. Leymann, "(2001) Web Services Flow Language," IBM Corporation, June 2002 Available from http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.

[4] K. Sriram, W. Patrick, P. Gregor L, "GSFL: A Workflow Framework for Grid Services," Preprint ANL/MCS-P980-0802 2002.

[5] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," IEEE Computer, January 2003.

[6] Workflow Management Coalition, "The Workflow Reference Model," Workflow Management Coalition, January 1995.

[7] G. Alonso, B. Reinwald, "Distributed Data Management in Workflow Environments," IEEE Trans. On RIB'97 1997.

[8] S. Silverberg, M. Beizer, "Microsoft MAPI Workflow Framework Concepts and Facilities," Microsoft Corporation Wang Laboratories, Inc. February 1996.

[9] Object Management Group, "Workflow Management Facility," April 2000, Available from ftp://ftp.omg.org/pub/docs/formal/00-05-02.pdf.

[10] K. Kim, C. Ellis, "Performance Analytic Models and Analyses for Workflow Architectures," Information Systems Frontiers, Kluwer Academic Publishers, September 2001.

[11] G.A. Bolcer, G. Kaiser, "SWAP: Leveraging the Web to Manage Workflow," IEEE Internet Computing, Vol.3, No. 1, January-February 1999.

[12] Workflow Management Coalition, "Workflow Standard-Interoperability Internet Wf-XML Binding," Workflow Management Coalition, July 1998.

[13] D. Barbara, S. Mehrota, and M. Rusinkiewicz,

"INCAs: Managing Dynamic Workflows in Distributed Environments," Journal of Database Management, 7(1):515, Winter 1996.
[14] R.S. Filho, J. Wainer, E.R.M. Madeira. C. Ellis, "CORBA Based Architecture for Large Scale Workflow," IEEE/IEICE Special Issue on Autonomous Decentralized Systems of the IEICE Transactions on Communications, Tokyo, Japan, Vol. E83-B, No. 5. 988-998. May 2000.
[15] 허원창, 배준수, 강석호, "비즈니스 프로세스 자동화를 위할 실행 모델의 설계", 한국경영과학회/산업공학회 춘계공동학술대회 2003년 5월.
[16] BEA, IBM, and Microsoft, "BPEL4WS Version 1.0," August 2002, available at http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/
[17] G. Laszewski, K.Amin, M. Hategan, B. Alunkal, S. Hampton, and A. Rossi, "GridAnt: Client-Side Workflow Management for Grids," 2003 Midwest Software Engineering Conference, June 2003.
[18] R. Kurmanowytsch, E. Kirda, C. Kerer and S. Dustdar, "OMNIX: A topology independent P2P middleware," UMICS 2003 Ubiquitous Mobile Information and Collaboration Systems Klagenfurt/Velden, Austria, June 2003.
[19] "Gnutella," The Gnutella homepage, available at http://gnutella.wego.com/
[20] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System," ICSI Workshop on Design Issues in Anonymity and Unobservability, LNCS 2009, July 2001.
[21] "The FastTrack Protocol," available at http://www.fasttrack.nu/
[22] "SimPy: A Python-based simulation package," SourceForge.net, June 2003. available at http://simpy.sourceforge.net/

이 이 섭

1998년 서강대학교 이공대학 수학과 1990년 서강대학교 전자계산학과 석사 1998년 고려대학교 정보통신대학 컴퓨터학과 박사 수료. 1990년~1992년 삼성종합기술원 그룹CAE센터. 1993년~1995년 삼성SDS 정보기술연구소. 1996년~1999년 삼성경영기술대학 정보통신학과 조교수. 2000년~현재 삼성SDS 정보기술연구소. 관심분야는 워크플로우, 지식관리시스템, P2P

박 수 현

1988년 고려대학교 컴퓨터학과 이학사 1990년 고려대학교 전산학 이학석사 1998년 고려대학교 컴퓨터학과 이학박사 1990년~1999년 (주)LG전자 중앙연구소 선임연구원. 1999년~2001년 동의대학교 공과대학 소프트웨어공학과 교수. 2002년~현재 국민대학교 비즈니스IT학부 교수. 관심분야는 이동통신 시스템, Active Network, 워크플로우, 컴포넌트 기반 개발

백 두 권

정보과학회논문지 : 컴퓨팅의 실제
제 9 권 제 2 호 참조