분산 트랜잭션 워크플로우 모니터링 시스템의 설계 및 구현

민 준 기[†] · 김 광 훈[††] · 정 중 수[†††]

요 약

본 논문에서는 분산 트랜잭션 워크플로우 모니터링 시스템의 설계 및 구현에 관하여 기술한다. 워크플로우 인스턴스들의 실행 상태 관리를 주요 목적으로 하는 전통적인 워크플로우 시스템의 모니터링 기능은 워크플로우의 초대형화와 트랜잭션화를 특징으로 하는 분산형 트랜잭션 워크플로우 시스템에서는 워크플로우의 처리 상태 관리 뿐 만 아니라 구조적인 정보의 수집, 통계 정보 제공, 사용자들의 처리 상태 정보 등과 같은 다양한 부가적인 정보 처리 및 관리 기능을 제공하는 분산형 모니터링 서비스를 필요로 한다. 본 논문에서는 이러한 분산형 트랜잭션 워크플로우 시스템에서 필수적으로 요구되는 새로운 워크플로우 모니터링의 특징을 확장 정의하고, 이를 지원하는 웹 기반의 분산 트랜잭션 워크플로우 모니터링 시스템의 상세 설계 및 이의 구현 방안을 소개한다.

키워드 : 워크플로우, 모니터링 시스템

Design and Implementation of a Distributed Transactional Workflow Monitoring System

Jun-ki Min[†] · Kwang-hoon Kim[††] · Joong-soo Chung[†††]

ABSTRACT

This paper describes the design and implementation details of a distributed transactional workflow monitoring system. There have been prevalent research and development trends in the workflow literature - workflow systems tend to be completely distributed architectures to support very large-scale workflow applications on object-oriented and internet-based infrastructures. That is, the active (object), distributed (architecture), system-oriented (transaction), and large-scale (application) workflow systems are the key targets in terms of the research and development aspects. While the passive, centralized, human-oriented, and small/medium scale workflow systems are the typical instances of the traditional workflow systems. Unlike in the traditional (the client-server architecture) workflow systems, the workflow monitoring features should not be easily supported in the recent (the fully distributed architecture) workflow systems. At the same time, they need a set of additional monitoring features, such as gathering and displaying statistical (or overload status) information of the workflow architectural components dispersed on the internet. We, in this paper, introduce the additional workflow monitoring features that are necessarily required for the recent workflow systems, and show how to embed those features into a web-based distributed workflow system.

Key Words : Workflow, Monitoring System

1. 서 론

워크플로우 기술은 비즈니스 프로세스의 자동화 이슈의 등장과 더불어 관심이 집중되기 시작한 사무 정보 시스템과 데이터베이스 시스템의 통합 기술중의 하나이다. 워크플로우는 컴퓨터 프로그램을 기반으로 하는 액티비티의 집합으로 이루어진 비즈니스 처리 절차를 모델링 한 것으로, 각각의 액티비티는 워크플로우 엔진의 제어를 통해 자동적으로 수행될 수 있다. 워크플로우 관리 시스템은 이러한 워크플로우 모델을 정의하고, 분석하는 모델링 시스템과 이 모델에 따라 비즈니스 프로세스의 각 액티비티의 실행을 자동화하고, 모니터링하는 엔진부분으로 구성되는데, 전자를 빌드타임 컴포넌트[11]라고 하며, 후자를 런타임 컴포넌트[11]라고 한다.

최근 워크플로우 기술에서의 주요 연구개발 동향을 살펴보면, 다음의 두 가지의 현상으로 요약할 수 있다[6, 12, 13]. 하나는 워크플로우 모델의 복잡성이 급속도로 증가하고 있다는 점이고[8], 다른 하나는 워크플로우를 구성하는 액티버티 프로그램이 트랜잭션화 된다는 점이다[7]. 이러한 동향 변화는 워크플로우 런타임 컴포넌트의 분산형 구조에 많은 영향을 주었을 뿐만 아니라 워크플로우 실행 모니터링 기능의 근본적인 특성에 영향을 주게 되었다. 즉, 기존의 워크플로우 모니터링 기능은 실행중인 워크플로우 인스턴스의 실행 상태를 그래픽 인터페이스를 통해 제공하는 실행상태 중

† 준 회 원 : 안동대학교 정보통신공학 박사과정
†† 정 회 원 : 경기대학교 정보과학부 부교수
††† 정 회 원 : 국립 안동대학교 공과대학 전자정보산업학부 정교수
논문접수 : 2005년 6월 21일, 심사완료 : 2006년 12월 14일

심의 모니터링 서비스[1-3, 11]을 제공하지만, 워크플로우 모델의 복잡성 증가와 액티버티의 트랜잭션화를 특징으로 하는 새로운 분산형 워크플로우 시스템에서는 워크플로우 인스턴스의 실행 통계 및 분석과 이를 이용한 워크플로우 마이닝 기술과의 접목을 가능하게 하는 분산형 워크플로우 모니터링 서비스[12, 13]의 제공을 요구하고 있다.

따라서, 본 논문에서는 한우리T/플로우[4]의 웹 기반의 분산형 워크플로우 관리 시스템의 런타임 컴포넌트에 적용될 분산형 모니터링 서비스를 새로이 정의하고, 이를 제공하는 분산형 워크플로우 모니터링 시스템의 설계 및 구현에 관하여 기술한다. 즉, 본 논문의 주요 목적은 기존의 워크플로우 관리 시스템의 런타임 컴포넌트의 부분적 기능의 하나였던 모니터링의 기능을 분리 및 확장해서 독립적인 모니터링 전용 런타임 컴포넌트를 설계 및 구현하는데 있다. 이를 위한 논문의 구성은 다음 장에서 분산형 트랜잭션 워크플로우 모니터링 서비스의 기본 개념을 정의하고, 이어서 이러한 분산형 워크플로우 모니터링 서비스를 지원하는 한우리T/플로우 분산형 트랜잭션 워크플로우 관리 시스템에 대한 전체적인 시스템구조를 기반으로 하는 분산형 모니터링 서비스 전용 런타임 컴포넌트의 상세설계 및 구현 내용을 기술하고, 이 시스템의 사용 예를 일련의 실제 스트린들을 통해 설명한다.

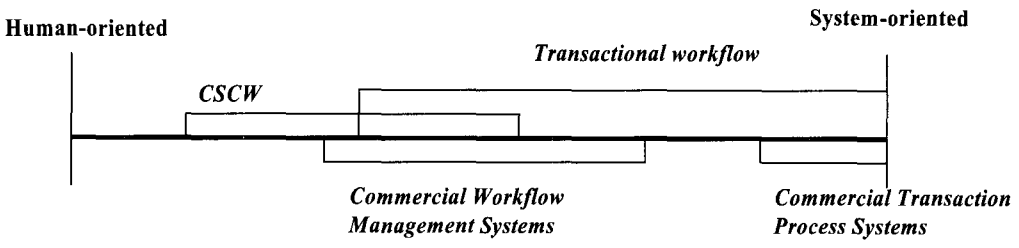
2. 관련 연구 및 배경

일반적으로 워크플로우 모니터링이라 함은 워크플로우 시스템에서 일어나는 이벤트를 추적하고, 기록하는 능력을 말한다. 워크플로우 모니터링 서비스는 시스템 수준의 모니터링부터 사용자 수준 혹은 응용 프로그램 수준의 모니터링 서비스까지 매우 다양하다. 시스템 수준의 전형적인 예를 들면 실행 요소들 즉, 분산 엔진등의 수행도나 적재량을 모니터 하는 것이다. 그리고, 워크플로우 인스턴스와 관련되서 각각의 액티비티의 상태를 보여주는 서비스는 전형적인 사용자 수준의 모니터링이라 한다. 이러한 서비스들은 워크플로우 실행 객체 간의 동작을 감지하는 부분과 실행 시간의 정보들을 접근하는 부분, 어떠한 변화가 일어날 때 마다 알릴 수 있는 부분을 통해 제공될 수 있다. 상호 운영의 부분에서 다시 살펴보면, 워크플로우 모니터링 서비스는 Push형태와 Pull형태로 나눌 수 있다[12]. 일반적으로, 거의 모든 시스템 수준의 모니터링 서비스는 Push형태로 동작을 한다. 반면, 사용자 수준의 모니터링 서비스는 Pull형태로 서비스를 제공한다.

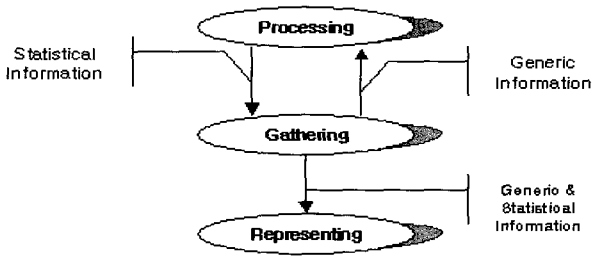
기존의 워크플로우 모니터링 서비스는 사용자 수준, Push형태로 제공하였다[8, 11]. 그러한 모니터링 서비스는 수동적, 상대적으로 간단한, 그리고, 인간 중심의 서비스이다. 또한, 기존의 워크플로우 시스템은 인간 중심의 분야 즉 주문처리 업무나 고용처리 업무와 같은 워크플로우 응용분야를 다루는데 이상적인 단일 시스템이었다. 이러한 시스템에서의 모니터링 서비스는 전통 모니터링 서비스 만으로도 충분히 그 역할을 담당했다. 그러나, 최근 분산 워크플로우 시스템과 트랜잭션 워크플로우 시스템에서는 전형적인 모니터링 서비스는 기능적인 면이나 구조적인 면에서 적당하지 않다. 그 기능은 확장되어야 하며, 분산 워크플로우 실행 구조를 고려해 재 설계를 해야 한다.

(그림 1)에서 볼 수 있듯이, 트랜잭션 워크플로우 시스템은 시스템 중심의 워크플로우 프로세스와 응용 분야를 지원해야 하기 때문에 전형적인 워크플로우 모니터링을 재설계해서 보다 효율적인 워크플로우 모니터링 기능을 제공해야 한다. 즉, 워크플로우 모니터링 시스템은 능동적, 시스템 중심, 분산 환경에 대응해야 한다. 특히, 인간 중심의 워크플로우 시스템에서 거의 모든 액티비티는 사람에 의해 조작이 되고, 작업이 종료하기 까지는 다소 시간이 걸린다. 이 의미는 액티비티의 상태 정보가 효율적으로 관리 되어야 하고, 사용자에게는 가치가 부여된다고 할 수 있다. 반면, 시스템 중심의 워크플로우 시스템에서 거의 모든 액티비티는 상대적으로 짧은 시간에 작업을 수행하는 프로그램이나 트랜잭션에 의해 진행된다. 이것은 모니터링 시스템이 액티비티의 상태정보 보다는 수행이 끝난 액티비티의 통계정보를 유지하는 것이 더 효율적이라 할 수 있다. 요컨데, 분산 워크플로우 구조와 트랜잭션 워크플로우 시스템의 추세에 따라 워크플로우 모니터링의 기능은 수동적, 인간중심, pull형태, 중앙집중형 구조, 상태정보 중심에서 능동적이고, 시스템 중심, push 형태, 분산구조와 통계정보 중심의 특징으로 변환되어야 한다[4, 12, 13].

(그림 2)는 모니터링 정보 처리부, 모니터링 정보 수집부, 모니터링 정보 표현부로 이루워진 분산형 트랜잭션 워크플로우 모니터링 서비스의 기능과 그들간의 상호 정보 교환을 설명한 것이다. 모니터링 정보 수집부는 워크플로우 실행부에게 정보를 요청하고, 상태정보나 이벤트 정보를 반환 받는 인터페이스를 가진다. 정보 처리부는 수집된 정보중 처



(그림 1) 분산형 트랜잭션 워크플로우 관리 시스템의 특징



(그림 2) 분산형 워크플로우 모니터링 서비스

리가 요구되는 정보를 처리하거나 분석하는 역할을 담당한다. 예를 들어 한 프로세스에 대한 통계 정보를 요구할 때에는 처리부에서 이를 처리하여 서비스를 하게 된다. 마지막으로, 정보 표현부는 모니터링 서비스를 받는 곳에서 쓰인다. 수집되거나 처리된 정보를 사용자에게 보다 인식되기 쉬운 방법으로 표현하는 역할을 담당한다. 예를 들어 통계정보의 표현은 단순한 문자 보다는 그래프 형태가 사용자에게는 인식하기가 쉽다. 이를 구체적으로 정의하면 다음과 같다.

• 수집부

이 특성은 트랜잭션 워크플로우 모니터링 시스템의 중요하면서 가장 기본이 되는 기능을 제공한다. 수집부는 수행중이거나 수행이 종료된 워크플로우에 대한 특정 정보의 집합을 수집한다. 그 정보는 다음과 같이 분류할 수 있다.

- 시스템
- 프로세스 인스턴스
- 액티비티 인스턴스
- 사용자

• 처리부

일반적으로 사용자는 통계정보와 같은 요약된 정보를 요구한다. 그래서, 수행부에 있는 인스턴스는 변화가 일어날 때 마다 그들의 상태나 이벤트를 특정 저장소에 저장을 해야한다. 또한, 동시에 수집부는 거대한 모니터링 정보를 처리부에 전송을 한다. 처리부는 저장소와 수집부로부터 정보를 전송받아 표현부에 표현할 수 있는 요약된 정보를 추출한다. 처리부에 의해 조작되는 정보는 특정 프로세스의 흐름정보, 위치정보, 프로세스나 액티비티의 수행도, 사용자와 시스템의 수행도등으로 이루어진다.

• 표현부

사용자는 복잡하고 거대한 정보를 좀더 정리되고 요약된 정보를 더 쉽게 이해할 수 있는 방법을 원한다. 예를 들어 마이크로 소프트의 윈도우 익스플로러와 같은 트리 형태로 인스턴스들을 리스트하거나 통계정보에 대해서는 막대 그래프를 이용하여 표현하는 것을 말한다. 표현부에서 가장 중요하게 다루워야 하는 부분은 프로세스 인스턴스에 대한 흐름을 보여주는 것이다. 이것은 플로우 다이어그램과 같은 형태로 보여 주어야 하고, 그 안에는 상태 및 진행상황을 나타내야 한다.

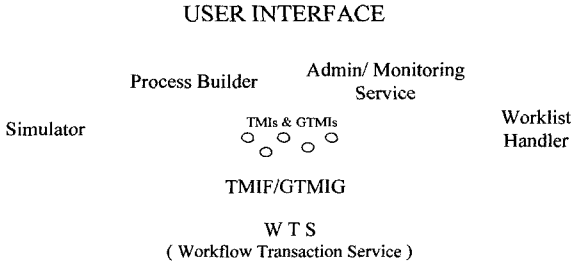
3. 분산 트랜재션 워크플로우 모니터링 시스템의 설계

본 장에서는 한우리/TFlow[4]의 시스템 구조를 소개하고, 이에 적용될 분산형 워크플로우 모니터링 시스템의 구조와 요구되는 정보를 정의하고, 이의 상세설계 내용을 기술한다.

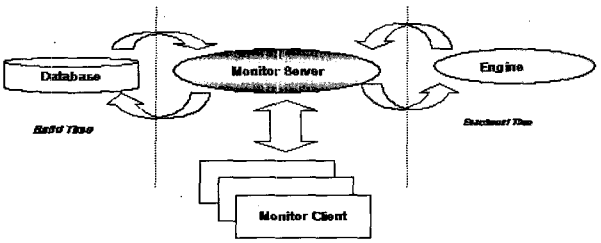
3.1 한우리/TFlow 시스템의 구조

분산형 트랜잭션 모니터링 서비스는 현재 개발 중의 워크플로우 관리 시스템인 한우리/TFlow의 한 모듈이다. 한우리/TFlow의 특징은 기능적 구조이고, 확장성, 유연성, 신뢰성, 웹 기반의 특성을 가진 워크플로우 관리 시스템이다. 한우리/TFlow는 크게 한우리T, 한우리 Flow 이 두가지 요소로 이루어져 있다. 한우리T는 한우리TP 모니터를 의미하는데, 이것은 워크플로우에서 트랜잭션의 개념을 말하며, JTS(java Transaction Services).를 포함한 OTS(Object Transaction Services)가 내장되어 확장된 한우리 TP 모니터를 근간으로 한다.

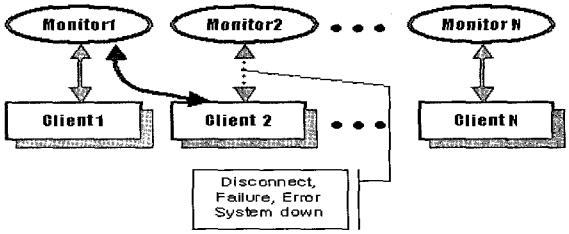
한우리T/Flow[4]의 엔진은 네 가지의 객체 GTMIG(Global Task Managing Instance Generator), TMIF(Task Managing Instance Factory), GTMI(Global Task Managing Instance) and TMI(Task Managing Instance)로 이루어져 있다. GTMIG는 전체 워크플로우 시스템의 중심으로 전체를 관장하는 역할을 하며, 하위 TMIF객체들을 관리하는 역할을 한다. 또한, TMIF에 인스턴스의 생성 명령을 내린다. TMIF는 GTMIG에서 내려온 명령을 수행하여 GTMI나 TMI를 생성하는 역할을 담당한다. 워크플로우 프로세스 인스턴스를 수행하는 객체가 GTMI이고, 액티비티 인스턴스를 수행하는 객체가 TMI이다. 따라서, 일반적인 워크플로우 시스템과 비교하여 "하나의 GTMI는 하나의 프로세스이고, 하나의 TMI는 하나의 액티비티"라 정의할 수 있다. 워크플로우 모니터링 서비스는 엔진의 객체를 신속히 추적해야 하므로 가까운 거리에 위치해야 한다. 즉, 엔진의 한모듈로 서존재하는 것이 바람직하다. 본 장에서는 실행부와 정의부 그리고, 모니터링부들간의 상호동작을 정의한다. 또한, (그림 2)에서 정의된 트랜잭션 워크플로우 모니터링 서비스의 기능을 설계한다.



(그림 3) 한우리/TFlow 시스템의 요소



(그림 4) 모니터링 시스템의 구조



(그림 5) 향상된 모니터링 이중화 시스템의 구조

3.2 분산 트랜잭션 워크플로우 모니터링 시스템의 구조

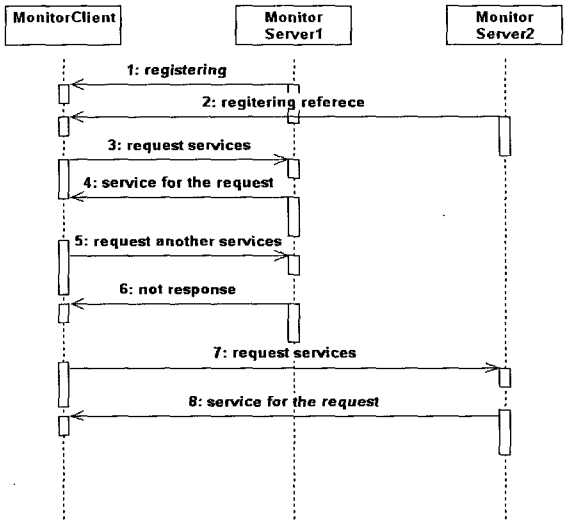
(그림 2)에서 언급한 분산 트랜잭션 워크플로우 모니터링 서비스는 워크플로우 엔진, 프로세스 빌더, 워크플로우 클라이언트 사이에 위치한다. 이것은 각각의 객체들과 연결하여 정보를 수집하고, 정보를 제공하는 것을 의미한다. 예를 들어, 워크플로우 모니터 클라이언트가 현재 진행되고 있는 프로세스의 정보나 진행된 끝난 프로세스의 통계정보를 요구한다며, 우선 모니터 서버에 연결하고, 모니터 서버는 그 요구를 충족시키기 위해 엔진과 데이터베이스를 연결하여 필요한 정보를 수집한다. 이 수집된 정보를 클라이언트에 서비스를 하는 구조이다. (그림 4)는 워크플로우 모니터링 시스템에 대한 전반적인 구조를 보여준다. 모니터링 시스템은 엔진이 분산 객체로 이루어져 있지만, 클라이언트-서버 구조를 가진다. 이것은 중앙 집중식으로 정보를 수집하고, 조작하는 것이 더 효율적이기 때문이고, 정보의 일치성(consistency)을 유지하여 준다. 따라서 모니터 서버는 가장 중요한 역할을 담당하기 때문에 신뢰성과 일치성, 가용성을 가져야 한다.

3.3 향상된 분산 트랜잭션 워크플로우 모니터링 시스템의 구조

위에 제시된 모니터링 구조의 큰 단점은 시스템이 대규모화 되고, 워크플로우 시스템에서 발생되는 인스턴스의 수가 증가하면, 과부하의 위험이 크다는 것이다. 또한 중심이 될 수 있는 모니터 서버가 자체적인 이유 뿐 아니라 외부적인 요인으로 인하여 통신이 단절 될 경우 치명적인 오류가 발생할 수 있다는 것이다. 따라서, 모니터링의 구조가 변화되어야 하는 것은 필수적이다.

(그림 5)의 향상된 모니터링 이중화 시스템 구조에서 알 수 있듯이 클라이언트 2는 모니터2에서 서비스를 받고 있는 도중, 네트워크의 문제나 모니터 서버 자체의 문제로 인해 더 이상 서비스를 받지 못하는 경우이다. 이 경우에 클라이언트 2는 모니터1에서 모니터링 서비스를 받을 수 있는 경로를 열어 주어야 한다. 이것은 모니터링 서비스의 가용성 향상을 위해 모니터 서버의 구조를 멀티 모니터 형태로 구성한 것이다.

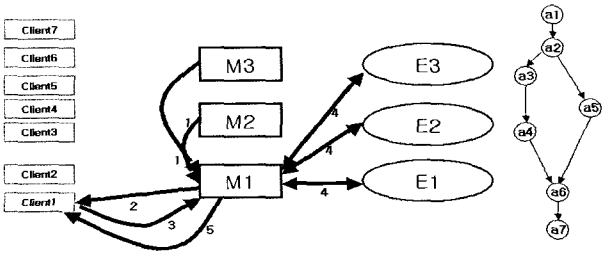
(그림 6)은 모니터링 서비스의 신뢰성과 가용성을 향상시키는 시나리오이다. 위에 나타낸 다이어그램은 모니터 클라이언트 하나와 모니터 서버 2가 있다고 가정을 한다. 모니터 클라이언트가 동작을 하게 되면 현재 가능한 모니터 서버의 레퍼런스를 수집한다. 모니터 서버에 우선 순위를 두어 우선 순위가 높은 모니터 서버에 먼저 연결을 하여 서비스를 받다가 도중에 서비스를 못 받는 경우에는 처음 구동 시 얻은 다른 모니터 서버에 연결을 하여 서비스를 받는 것이다.



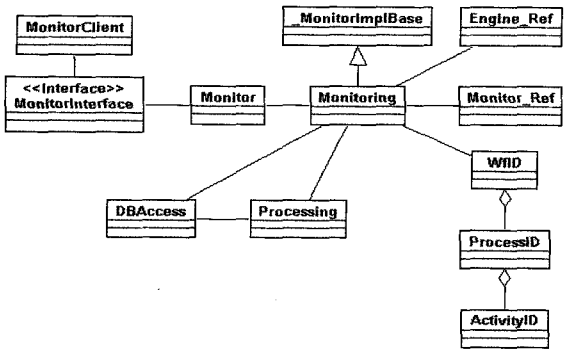
(그림 6) 가용성 향상을 위한 시나리오

본 장에서 제안된 모니터 서버의 구조는 클라이언트 서버 구조의 확장된 구조로 모니터서버를 여러 개 두어 서비스를 받게 할 수 있는 구조이다. 이는 서버 자체 부하에 크게 관심을 가져야 한다. 그러나 2장에서 언급 하였듯이 모니터링의 기능은 클라이언트의 요구를 충족 시키기 위한 것이다. 따라서 모니터 서버 자체에는 저장능력을 가질 필요가 없다. 다만 클라이언트의 요청이 많을 경우 병목 현상이 있을 수 있다. 향상된 모니터링의 구조는 위에 제시된 문제를 모두 해결하는 방법이다.

(그림 7)은 향상된 모니터링 시스템의 전체적인 구조이다. E는 엔진을 의미하며, M은 모니터 서버를 의미한다. 앞에서 언급한 시나리오를 이용하여 순서를 결정한 것이다. 결국,



(그림 7) 향상된 모니터링 시스템의 구조



(그림 8) 모니터링 서버의 클래스 다이어그램

이러한 구조가 모니터 서버의 신뢰성을 확보하고, 모니터 클라이언트에게 가용성을 향상 시킨 구조이다. 또한 자체 부하가 없고, 여러 개의 서버를 나누어 클라이언트의 병목 현상을 해소 하였다.

3.4 분산 트랜잭션 워크플로우 모니터링 서버 설계

- 모니터링 서버의 역할은 엔진과 데이터 베이스에 저장된 내용을 수집하여, 가공할 필요성이 있는 정보는 가공을 하여 서비스를 하는 것이다.
- 모니터링 서버의 구성은 전체적인 모니터링에 관련된 모듈인 Monitoring Class, 데이터베이스와의 연동[10]을 위한 DBAccess Class, 정보 가공을 위한 Processing Class, 엔진과 데이터 베이스에 있는 객체를 참조하기 위한 각 레퍼런스 클래스가 있고, 향상된 모니터링 시스템을 구성할 수 있는 모니터링 서버에 관한 레퍼런스 클래스로 구성된다. 이렇게 모니터링 서버를 구성하고 각 클래스는 표준화 기관에서 권고하는 API를 이용하여 다른 모듈과 통신을 한다.
- WfMC(Workflow Managemant Coalition)[1-3], OMG (Object Management Group)[5]에서 권고하는 모니터링을 위한 API로는 모니터 서버를 구성할 수 없다. 따라서 본 논문에서 기술되는 모니터링 서버는 모니터링을 위한 다양한 API를 제안한다. 트랜잭션 워크플로우를 위한 모니터링 서비스를 하기 위해서는 통계정보를 요구한다. 프로세스 관리자 수준에서 다음과 같은 API를 통해 이벤트에 관련된 정보를 수집하여 통계를 처리하게 된다.

1. lastEvent()

반환 값은 AuditEvent객체이고, 마지막으로 발생된 이벤트를 감시하기 위한 API이다.

2. getIterator()

반환 값은 AuditEvent의 Iterator객체이고, 모든 이벤트를 차례로 저장된 형태이다.

3. getEventsIterator(in string QUERY)

반환 값은 AuditEvent의 Iterator객이고, 조건에 부합하는 이벤트를 추출한다.

4. howManyEvents(in string QUERY)

반환 값은 정수 형이고, 이벤트의 개수를 추출할 때 사용된다.

이러한 메소드를 이용하여 정보를 수집하고, 각 프로세스 상태 조건으로 통계정보를 추출한다. 통계정보는 일반적으로 수량과 시간에 대해 추출을 한다.

3.5 분산 트랜잭션 워크플로우 모니터링 클라이언트 설계

모니터링 클라이언트는 두 가지로 볼 수 있다. 하나는 워크플로우 클라이언트를 위한 모니터링 클라이언트이고, 다른 하나는 운용 관리 도구를 위한 모니터링 클라이언트이다. 다시 말하면, 워크플로우 관리자를 위한 모니터링과 워크플로우 일반 사용자를 위한 모니터링으로 나누어서 설계를 해야 한다. 먼저, 워크플로우 관리자는 시스템에 관한 사항을 서비스 받기를 원한다. 구체적으로 나타내면, 시스템에서 생성된 객체의 수나 어떤 프로세스가 시스템의 영향에 의해 에러가 발생했는가에 대한 정보이다. 반면에 워크플로우 사용자는 시스템 보다는 자신이 속한 워크플로우 프로세스의 진행에 더욱 관심이 있다. 그 프로세스가 어느정도 진행이 되었고, 앞으로의 예상 종료 시간등을 요구한다. 그러나 워크플로우 사용자에 대한 모니터링 서비스는 관리자 정보의 부분 집합으로 포함되어 있기 때문에 본 논문에서는 관리자를 위한 모니터링 서비스에 대해서만 언급을 한다.

워크플로우 모니터 클라이언트는 모니터링 서버에서 수집되거나 처리된 정보를 서비스 받아 화면에 보여주는 역할을 담당한다. 이에 모니터 클라이언트의 화면에 보여주는 정보는 전체적인 시스템, 프로세스, 액티비티와 그에 대한 인스턴스의 모든 레퍼런스와 각각에 대한 속성값, 또한 현재 진행 상태, 시스템 수행도, 통계정보이다. 이러한 정보를 다음과 같은 특징으로 표시한다.

- 트리 : 워크플로우 시스템, 프로세스, 액티비티, 그에 대한 인스턴스를 표현하기 위해 사용된다. 트리는 정의부와 실행부로 나뉘어 보여지는데, 정의부는 데이터베이스에 정의된 프로세스와 액티비티를 나타내고, 실행부는 엔진에서 진행중인 프로세스 인스턴스와 액티비티 인스턴스, 시스템에 대한 레퍼런스로 구성된다.
- 표 : 트리에서 나타난 레퍼런스의 구체적인 속성값을 나타낼 때 사용된다.
- 플로우 다이어그램 : 현재 진행 상태를 표시할 때 사용된다. 이는 워크플로우 정의부에서 정의된 그림과 동일하지만, 액티비티를 나타내는 아이콘이 현재의 상태를 나타내는 아이콘으로 구성된다. 각각의 아이콘이 현재 상태를 나타내기 때문에 사용자는 빠른 시간 내에 이해를 할 수 있다. 그 아이콘은 (그림 9)와 같이 정의한다. 이 아이콘의 두 가지의 특징으로 구성되어 있는데, 하나는 기어모양의 위치, 다른 하나는 색깔이다. 기어모양의 위치가 아래에 있다는 의미는 실행이 아직 안된 상태를 의미하고, 위로 올라 갈수록 종료에 가까운 상태

(그림 9) 상태 아이콘 정의

이다. 또한 초록 색깔은 정상 동작을 의미하며, 빨간색으로 갈수록 정상 동작을 하지 않는 상태를 의미한다.

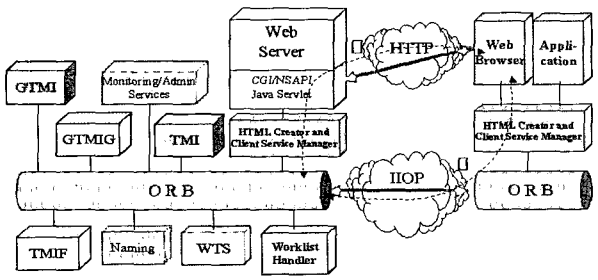
- 그래프 : 통계 정보와 시스템의 수행도를 나타낸다. 통계 정보는 사용자가 인식하기 편하게 막대 그래프와 파이 그래프를 이용하여 프로세스를 평가할 수 있는 자료를 마련하고, 선 그래프를 이용하여 시스템의 가지고 있는 객체에 대해 모니터링을 하여 과부하를 줄일 수 있는 정보를 제공한다.

4. 분산 트랜잭션 워크플로우 모니터링 시스템의 구현

본 장에서는 한우리/TFlow[4] 시스템과 모니터링 서비스의 실제적인 구현에 관하여 기술하고, 시스템의 환경과 클라이언트의 동작 관계를 보여준다.

4.1 시스템 운용 환경

한우리/TFlow 시스템은 CORBA(Common Object Request Broker Architecture)[8]환경에서 자바로 구현되었다. 그것은 웹 사용자 인터페이스를 제공하며, 플랫폼 뿐 아니라 응용 프로그램에 독립적이다. 모든 객체는 ORB를 통해서 통신을 하고, 워크플로우 사용자는 HTTP를 이용하여 접속한 다음 IIOP를 통해 서비스를 받는다(그림 10). 먼저 모든 객체의 레퍼런스를 관리하는 Naming Services를 동작하고, TMIF와 GTMIG를 구동시킨다. 다음 워크리스트 핸들러, 운용 관리, 모니터 서버를 구동시킨다. 워크플로우 사용자는 브라우저를 이용하여 웹을 통해 웹 서버에 접속을 한 다음 워크플로우 클라이언트를 다운로드 받게 된다. 그러면 간단한 인증 절차를 통해 워크플로우 작업 할당을 받고 작업을 개시하게 된다. 이러한 상황에서 워크플로우 사용자가 모니터링 서비스를 받기를 원할 때 간단한 마우스의 클릭으로 현재 진행상황등 모니터링 서비스를 받게 되는 것이다. 여기서 트랜잭션에 관련된 부분은 WTS가 담당하게 된다.



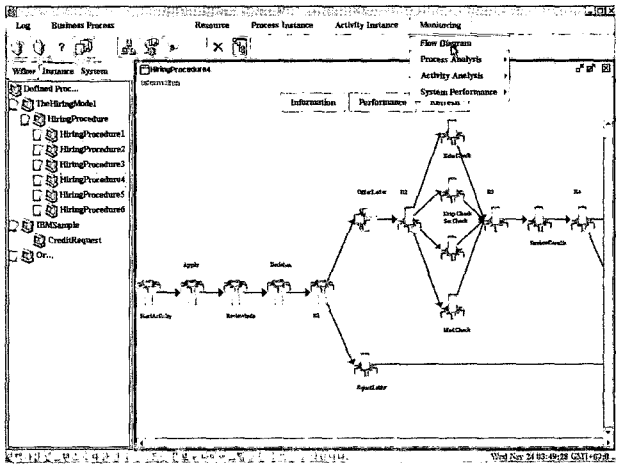
(그림 10) 한우리/TFlow 시스템의 운용 환경

4.2 운용 예

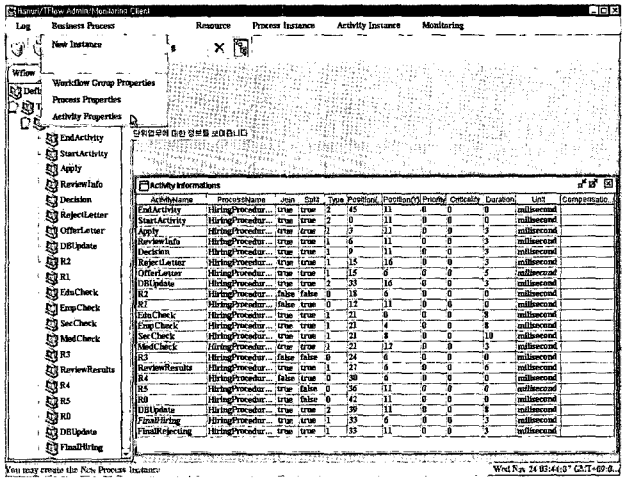
분산 트랜잭션 워크플로우 모니터링 시스템의 구현에 관한 일련의 스크린을 통해 운용 예를 소개하고자 한다. 적용된 예제는 IBM 회사의 고용 워크플로우 모델로서, 피고용자가 접수를 하면서 고용 프로세스는 시작이 되는데, 여러 가지의 절차를 거쳐 합격, 불합격의 판정이 내려 진다.

(그림 11)은 한 프로세스에 관한 진행 상태를 보여주는 플로우 다이어그램이다. 현재 6번째 액티비티가 진행되고 있는 상황을 볼 수 있다. 오른쪽 부분의 트리가 워크플로우 객체들에 대한 이름들로 구성된다. 이는 모든 명령을 내리기 위한 이름으로 중요한 부분이다.

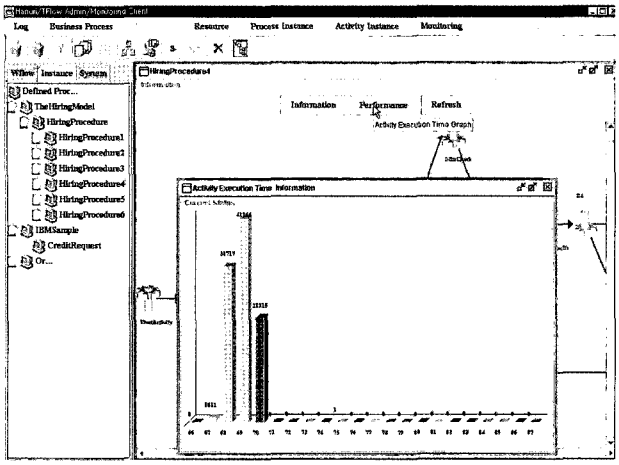
다음 (그림 12)는 한 프로세스에 속한 액티비티들의 속성값을 자세히 보여주는 그림이다. (그림 13)은 통계를 정보를



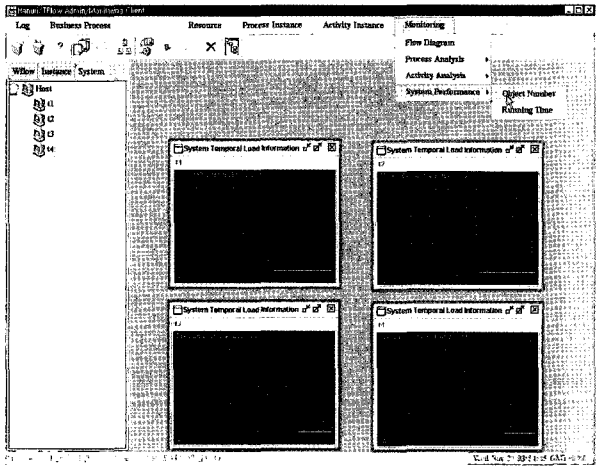
(그림 11) 상태 흐름 다이어그램



(그림 12) 프로세스의 속성 테이블



(그림 13) 프로세스 통계 정보(막대그래프)



(그림 14) 분산 모니터링 서버들의 실행 및 부하 상태

나타내는 막대 그래프이고, 각 액티비티의 진행 시간을 나타낸다. (그림 14)는 시스템의 수행도를 나타내는데, 현재 4개의 시스템에서 생성된 객체의 수를 실시간으로 화면에 보여준다. 이는 관리자 시스템에 대한 모니터링 중 한 시스템의 과부하를 줄일 수 있는 정보를 제공한다.

5. 결 론

본 논문은 인터넷 상에서 동작하는 웹 기반의 사용자 인터페이스를 사용한 분산 트랜잭션 워크플로우 모니터링 시스템의 설계 및 구현에 관하여 기술하였다. 구체적으로 분산 트랜잭션 워크플로우 시스템에서 요구되는 워크플로우 모니터링 서비스의 부가적인 기능과 특징을 재정의하였다. 분산 트랜잭션 모니터링 시스템에 요구되는 사항은 기존에 모니터링 서비스에서 제공하는 상태정보 이외에 다양한 통계정보와 시스템 정보를 제공하는 것이다. 분산 트랜잭션 워크플로우 시스템은 거의 대부분의 절차가 트랜잭션이나 소프트웨어 프로그램으로 구성되어 그 작업 시간이 짧기 때문에 진행 상태에 대한 정보는 의미가 없다. 따라서 이러한 기존의 워크플로우 모니터링 개념을 확장하여, 그 기능을 재정의 하고 설계 및 구현을 하였다. 또한, 모니터링 시스템의 일치성과 가용성, 신뢰성을 향상시키기 위한 분산 모니터링 시스템의 구조를 제안하였고, 구현함에 있어 표준기관의 권고안에서 부족한 API를 정의하였다. 향후의 연구 방향은 적응형 워크플로우(Adapative Workflow)에서의 워크플로우 모니터링의 요구사항을 추출하여 정의하는 것이다. 또한, 페트리 넷에서 사용되는 "Firing Sequence"의 개념을 도입해 수행이 된 워크플로우 프로세스를 분석하고, 테스트할 수 있는 워크플로우 마이닝 기술로의 확장을 시도해야 할 것이다.

참 고 문 헌

[1] "Workflow Management Coalition Specification Document: The Workflow Reference Model," WfMC Document Number TC00-1003, Jan., 1995.

[2] "Workflow Management Coalition Specification Document: Audit Data Specification," WfMC Document Number WFMC-TC-1015, Sep., 1998.

[3] "Workflow Management Coalition Specification Document: Terminology & Glossary," WfMC Document Number WFMC-TC-1011, June, 1996.

[4] Kwang-Hoon Kim, Dong-su Han, Moon-Ja Kim and Young-Cheol Ryoo, "Hanuri/TFlow : A Transactional Workflow Management System For Cross-Organizations," Kyonggi Univ., Information and Communication Univ., ETRI, Feb., 1999.

[5] OMG BODTF RFP #2 Submission Workflow Management Facility Revised Submission, OMG Document Number bom/98-06-07.

[6] Jari Venijalainen, Aarno Lehtola, Olli Pihlajamaa, "Research Issues in Workflow System," VTT Information Technology, Oct., 1995.

[7] "Transaction Workflow Management System Requirements Specification," ETRI, Sept., 1998.

[8] J.A. Miller, A.P. Sheth, K.J. Kochut and X. Wang, "CORBA-Based Run-Time Architectures for Workflow Management Systems," Large Scale Distributed Information Systems Lab, Department of Computer Science, The University of Georgia, Technical Report, 1998.

[9] Robert Orfali, Dan Harkey "Client/Server Programming with Java and CORBA", Wiley, Second Edition.

[10] Graham Hamilton, Rick Cattell, Maydene Fisher "JDBC Database Access with Java", ADDISON WESLEY.

[11] Diimitrios Georgakopoulos and Mark Hornick "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure", Distributed and Parallel Databases, 3, 119-153(1995), GTE Laboratories

Incorporated. Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

[12] H. S. Hong, et al., "A Web-based Transaction Workflow Monitoring System," Proceedings of the International Conference on WISE, IEEE PRESS, May, 2000.

[13] Kwanghoon Kim and Hyungjin Ahn, "An EJB-based Very Large Scale Workflow System and Its Performance Measurement," Proceedings of the 6th International Conference on Web-age Information Management, Lecture Notes in Computer Science, Oct., 2005.

민 준 기

e-mail : jakimin@tta.or.kr

1993년 한남대학교 대학원 전자계산학과 (석사)

2003년~현재 국립 안동대학교 정보통신공학 박사과정

1993년~2003년 우송대, 대덕대, 혜천대, 경기대, 경기대 정보통신대학원 시간강사

1980년~1999년 ETRI

1999년~2001년 LOCUS

2001년~현재 TTA

관심분야 : 데이터네트워크, 멀티미디어, BPM, Workflow, eBiz, etc.,

김 광 훈

e-mail : kwang@kyonggi.ac..kr

1986년 중앙대학교 대학원 전자계산학과 (석사)

1994년 University of Colorado at Boulder, Computer Science, MS

1998년 University of Colorado at Boulder, Computer Science, Ph.D

1986년~1991년 ETRI

1993년~1994년 American Educational Products Inc. Professional DB Consultant

1994년~1995년 Colorado Advanced Software Institute. Research Assistant

1995년~1997년 Azt다 Engineering Inc, Software Engineer

1998년~현재 경기대학교 정보과학부 부교수

관심분야 : 워크플로우, 그룹웨어, 컴퓨터네트워크, 데이터베이스.

정 중 수

e-mail : jschung@andong.ac.kr

1983년 연세대학교 전자공학과(석사)

1993년 연세대학교 전자공학과(박사)

1983년~1994년 ETRI

1987년~1989년 벨지움 Alcal/Bell Telephone사 객원연구원

2000년~2001년 미국 UMASS/Lowell 전산학과 객원교수

1994년~현재 국립 안동대학교 공과대학 전자정보산업학부 부교수

관심분야 : 데이터네트워크, 멀티미디어, BPM, etc