분산 스케쥴링을 이용한 이동 에이전트 기반 워크플로우 시스템의 설계

박원주° 김병만 김현수
금오공과대학교 컴퓨터공학과
{wjpark, bmkim, hskim}@cespc1.kumoh.ac.kr

Design of Mobile Agent based-Workflow System using Distributed Scheduling

Won Ju Park° Byeong Man Kim Hyeon Soo Kim
Dept. of Computer Engineering, Kumoh National University of Technology

요 약

기존 워크플로우 시스템은 비즈니스 프로세스 내의 정보와 제어의 흐름 수행에 있어서 중앙 집중적인 워크플로우 엔진이 전체 실행의 모든 부분을 담당하고 있다. 이러한 워크플로우 시스템이 갖는 성능, 확장성 등 구조적 제약의 단점을 극복하기 위하여 워크플로우 엔진의 분산, 이동 에이전트의 적용, 웹 기반 등 다양한 연구가 진행되고 있다. 본 논문에서는 단위 업무 스케쥴링 기능과 관련된 정보를 분산된 형태로 참가자에게 일임하여 워크플로우 엔진 부하를 줄이고, 프로세스 인스턴스 단위의 이동 에이전트를 적용하여 비동기적 특성을 이용한 성능 및 확장성의 향상을 제공하는 워크플로우 시스템의 설계를 제안한다. 또한 제안한 시스템을 위하여 각 참가자별 스케쥴링 정보를 WfMC의 프로세스 정의 표준 언어인 WPDL로부터 추출하는 방법을 제안한다.

1. 서론

워크플로우란 절차적인 규칙들의 집합에 의한 전체 또는 부분적인 비즈니스 프로세스의 자동화를 의미하고, 워크플로우 엔진은 비즈니스 프로세스 수행을 위해 프로세스 정의 해석, 워크플로우 참가자와의 상호 작용, IT 도구와 응용 프로그램 호출 등의 역할을 수행한다. 그리고 워크플로우 관리 시스템이란 하나 이상의 워크플로우 엔진을 통하여 워크플로우 정의, 생성, 실행 관리를 담당하는 시스템을 뜻한다[1]. 1980년대 초 워크플로우 개념이 처음으로 등장한 이래, 기업 조직에 있어 급변하는 기업 환경 변화 대처 및 기업 경쟁력 확보를 위한 워크플로우 시스템의 요구는 계속 증가되어왔다. 현재 워크플로우 시스템은 단순한 업무 자동화가 아닌 비즈니스 프로세스를 기반으로 기업 내 다양한 IT 시스템들과 연동 가능한 상호 운용성을 갖춘 IT 통합으로서 역할이 요구된다. 그러나 현존하는 워크플로우 시스템은 시스템간 호환성, 중앙 집중형 구조로 인한 성능, 확장성, 견고성과 가용성 등의 한계를 지니고 있다. 이러한 문제 해결을 위해 워크플로우 실행의 분산화, 비동기적 수행을 위한 이동성, 트랜잭션 지원, 복제 등이 요구된다[2]. 위 요구사항을 만족하기 위해 워크플로우에 대한 표준화 노력과 분산 기반, 이동 에이전트의 적용, 트랜잭션 적용, 웹 기반 등의 기술 적용에 대한 연구가 진행 중이다[3]. 본 논문에서는 위와 같은 문제점을 해결하기 위해 단위 업무 스케쥴링 기능과 관련된 정보를 분산된 형태로 참가자에게 일임하여 워크플로우 엔진 부하를 줄이고, 프로세스 인스턴스 단위의 이동 에이전트를 적용하여 비동기적 특성을 이용한 성능 및 확장성의 향상을 제공하는 워크플로우 시스템의 설계를 제안한다.

2. 관련 연구

2.1 WfMC(Workflow Management Coalition)

WfMC는 워크플로우 관리 시스템간 상호 운용성과 워크플로우 제품간의 호환성 문제를 해결하기 위해 모인 워크플로우 표준화 단체이다. 이를 위해 워크플로우 구성 요소와 구성 요소간 표준화된 인터페이스를 정의한 워크플로우 참조 모델을 제시하였다[4]. 그 중 인터페이스 1은 프로세스 정의를 기술하기 위한 공통의 메타 모델을 포함하며, 이는 WPDL(Workfow Process Definition Language)로 표현된다. 본 논문에서는 프로세스 정의에 사용되는 WPDL로부터 참가자 별 스케쥴링 정보를 추출하여 각 참가자에 대한 스케쥴링 정보를 프로세스 테이블로 구성하는 방법을 제안한다.

2.2 에이전트 기반 워크플로우 시스템들

Dartmouth 대학의 DartFlow는 유연성, 적응성, 에러 처리, 확장성을 목표로 한 이동 에이전트 기반, 웹 가능 워크플로우 시스템이다. 웹 브라우저를 사용자 인터페이스로 제공하고, 백본은 Agent-TCL을 이용하여 워크플로우를 처리한다[5]. DartFlow에서 하나의 프로세스 인스턴스는 하나의 프로세스 에이전트에 의해 수행된다. 프로세스 에이전트 수행 중 참조하는 스케쥴링 정보는 에이전트 서버에 의해 제공되고, 프로세스 에이전트는 이러한 정보를 이용하여 자신의 워크플로우를 진행한다. 이는 집중화된 에이전트 서버 특성상 프로세스 인스턴스가 많을 경우 에이전트 서버의 병목 현상이 발생 할 수 있으며 매 단위 작업 수행 후 작업 리스트 서버로의 접속 시 병목 현상을 초래할 수 있다. 다른 연구로서는 워크플로우 시스템 내 워크플로우 정의를 해석, 제어하는 스케쥴링 모듈을 별도로 두어 워크플로우 엔진과 에이전트 서버와의 적절한 기능 분할을 통한 워크플로우 엔진의 부하 감소를 목적으로 하는 시스템이 있다[6]. 이 연구에서는 하나의 프로세스 인스턴스를 담당하는 프록시 에이전트는 처리해야 할 과업을 스케쥴링 모듈로부터 받고, 각 과업의 수행은 프록시 에이전트가 생성한 서브 에이전트에 의해 이뤄진다. 그러나 분기 제어 시점에서는 처리해야 할 과업을 받는 과정에서 집중화된 스케쥴링 모듈의 처리를 기다리는 병목 현상이 발생할 가능성이 존재한다.

3. 제안한 워크플로우 시스템의 설계

본 논문에서 제안한 분산 스케쥴링을 이용하는 이동 에이전트 기반 워크플로우 시스템은 하나의 워크플로우 프로세스 인스턴스 발생 시 하나의 프로세스 에이전트가 생성되어 해당 인

스턴스에 대한 데이터를 소지하며 프로세스에 관여하는 참가자들을 거쳐 주어진 프로세스 인스턴스를 마치게 된다. 각 참가자마다 관련된 스케쥴링 정보는 프로세스 테이블로 존재하고, 기존 워크플로우 엔진이 수행하던 프로세스 실행 제어 스케쥴링 수행 부분은 참가자 측에 존재하는 참가자 에이전트에 의해 이뤄진다. 제안하는 시스템의 구조는 그림 1과 같다.

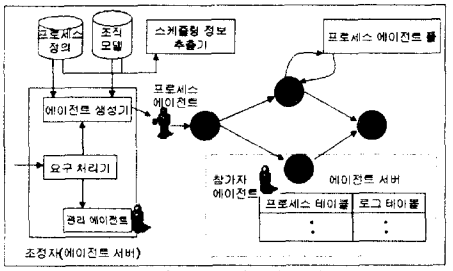


그림 1. 워크플로우 시스템 구조

3.1 시스템 구성 요소

1) 조정자

조정자는 에이전트 서버를 기반으로 하며 사용자의 요구를 처리한다. 조정자는 아래와 같은 구성 요소로 이뤄져 있다.

① 요구 처리기

프로세스 인스턴스의 시작, 관리·감독의 요구를 받아 에이전트 생성기 및 관리 에이전트에게 넘겨준다.

② 에이전트 생성기

하나의 프로세스 인스턴스를 담당하는 프로세스 에이전트를 생성한다. 생성된 에이전트는 초기화 과정을 거친 후 에이전트 서버의 도움을 받아 초기 시작 참가자의 에이전트 서버로 파견된다.

③ 관리 에이전트

관리·감독 작업을 수행하는 이동 에이전트로서 각 참가자 측에 저장된 로그 테이블을 참조하여 프로세스 에이전트를 추적한다.

2) 프로세스 에이전트

에이전트 생성기로부터 생성된 프로세스 에이전트는 각 참가자 컴퓨터로 파견되어 참가자 에이전트의 도움을 받아 스케쥴링 되어 진행된다. 프로세스 에이전트는 프로세스 인스턴스에 관계된 데이터와 워크플로우 관계 데이터 등 프로세스 인스턴스 수준의 데이터를 가진다. 또한 에이전트의 자율적인 특성을 이용하여 워크플로우 제한 시간 등의 예외 상황에 능동적으로 대처할 수 있다.

3) 참가자 에이전트

워크플로우 참가자는 워크플로우 참가자 컴퓨터에 존재하며 참가자가 수행해야 할 단위 작업 처리와 프로세스 에이전트에 대한 스케쥴링 작업을 수행한다. 또한 관리 에이전트를 도와 관리·감독 기능을 수행한다. 각 참가자가 수행해야 할 단위 작업에 대한 정보 및 스케쥴링 정보는 참가자마다 별도의 프로세스 테이블로 저장된다. 프로세스 테이블은 프로세스 종류별로 따로 관리되며 구성 레코드 필드 내용은 아래와 같다.

프로세스 종류					
이전 참가자	진입 분기	단위 작업 종류	진출 분기	전이 조건	다음 참가자

표 1. 프로세스 테이블 레코드 필드

이전 참가자와 이후 참가자 필드는 참가자의 ID와 위치가 기록된다. 참가자의 수는 조건에 따라 여러 명이 될 수 있다. 진입, 진출분기는 WPDL에 기술된 분기를 기술한다. 각 분기의 자세한 사항은 3.3절에서 다룬다. 단위 작업의 종류는 WPDL에 기술된 NO(참가자 직접 수행), APPLICATIONS(응용 프로그램 지원), WORKFLOW(다른 서브 프로세스 수행), LOOP(다른 단위 작업들의 루프 수행), ROUTE(조건 분기 작업 용도)으로 나뉜다. 단위 작업이 WORKFLOW, LOOP, ROUTE인 경우는 실제 참가자가 없는 경우이므로 시스템 내 별도의 참가자를 두어 수행한다. 전이 조건은 현재 참가자에서 다음 참가자로 프로세스 에이전트를 파견하기 위한 스케쥴링 조건을 기술한다. 또한 참가자 에이전트는 관리 에이전트와의 관리·감독 작업을 위해 단위 작업 처리 로그, 자신의 프록시 등을 내용으로 하는 로그 테이블을 유지한다. 로그 테이블의 레코드는 표 2와 같다.

로그 테이블					
프로세스 ID	수행 상태	프로세스 에이전트	참가자 에이전트	다음 참가자	필드
프로세스 에이전트가 수행하는 프로세스 인스턴스 ID	1. 대기 2. 수행 중 3. 파견 4. 실패 5. 삭제	프록시 저장	프록시 저장	프로세스 에이전트가 다음 참가자로 파견된 경우 기술	설명

표 2. 로그 테이블 레코드 필드

4) 프로세스 에이전트 풀

워크플로우 시스템 내 별도의 컴퓨터에 존재하며 다음 참가자의 컴퓨터가 꺼져 있을 때 워크플로우 시스템에 다시 참가할 때까지 수행 중인 프로세스 에이전트를 저장하기 위한 임시 저장소 역할을 한다. 참가자가 자신의 위치를 변경한 후 다시 워크플로우 시스템에 참가할 때는 참가자 에이전트의 도움을 받아 프로세스 에이전트 풀에 메시지를 보내어 자신이 수행해야 할 프로세스 에이전트를 파견 받은 후 할당된 단위 작업을 계속 수행하고, 자신의 프로세스 테이블 필드 중 이전, 이후 참가자에게 자신의 변경된 위치를 알린다.

3.2 시스템 수행 과정

제안된 워크플로우 시스템의 수행 과정은 다음과 같이 이뤄진다. 초기 프로세스 인스턴스 생성은 사용자가 요구처리기에게 원하는 프로세스 타입과 인스턴스 데이터를 보냄으로서 시작된다. 에이전트 생성기는 요구된 프로세스 종류에 맞는 프로세스 에이전트를 생성하고 인스턴스 데이터, 프로세스 ID를 할당하고 워크플로우 관계 데이터를 초기화한다. 이후 프로세스 에이전트는 처음 참가자로 파견되어 로그 테이블에 로그를 남긴 뒤 참가자 에이전트에게 처리를 맡긴다. 참가자 에이전트는 프로세스 에이전트가 갖는 동일한 프로세스 종류의 프로세스 테이블을 참조하여 진입 분기와 이전 참가자 사항을 비교한다. 진입 분기가 AND_JOIN인 경우는 기술된 이전 참가자로부터 동일한 프로세스 ID의 프로세스 에이전트들이 도착한 후 주어진 단위 작업을 초기화한다. XOR_JOIN인 경우는 이전 참가자로부터 프로세스 에이전트 도착 시 바로 단위 작업을 초기화한다. 단위 작업을 마친 후에는 참가자 에이전트는 진출 분기와 전이 조건을 평가하여 다음 참가자를 결정하고 이를 프로세스 에이전트에게 알린다. 이에 프로세스 에이전트는 포함하는 이전 참가자 정보를 현재 참가자 정보로 수정한 후 참가자 에이전트에 의해 다음 참가자로 파견된다. 파견된 후에는 로그 테이블의 다음 참가자 항목이 기술된다. 진출 분기가 AND_SPLIT인 경우는 프로세스 에이전트를 복제하여 전이 조건에 맞는 모든 다음 참가자들에게 파견하고 XOR_SPLIT인 경우는 한 참가자에게만 파견한다. 위 과정을 반복함으로서 하나의 프로세스 에이전트의 수행을 마치게 된다.

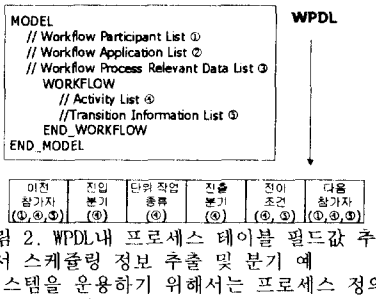


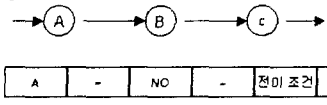
그림 2. WPDL내 프로세스 테이블 필드값 추출

3.3 WPDL에서 스케쥴링 정보 추출 및 분기 예

제안한 시스템을 운용하기 위해서는 프로세스 정의로부터 각 참가자들이 갖는 스케쥴링 정보를 위한 프로세스 테이블 구성

이 중요하다. 그림 2는 프로세스 테이블을 구성하기 위해 WfMC 표준 프로세스 정의 기술 언어인 WPDL(Workflow Process Definition Language)에서 참조하는 부분을 나타낸다. 그리고 진입, 진출 분기에 따른 프로세스 테이블은 아래 예와 같이 구성된다.

① 순차 진행

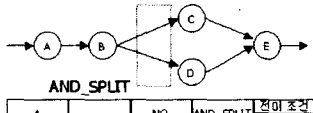


A	-	NO	-	전이 조건	c

그림 3. B 참가자의 프로세스 테이블

전이 조건이 존재하면 조건 평가 후 참인 경우에만 프로세스 에이전트는 다음 참가자에게로 파견된다.

② AND_SPLIT

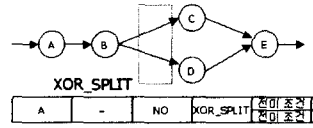


A	-	NO	AND_SPLIT	전이 조건	C
				전이 조건	D

그림 4. B 참가자의 프로세스 테이블

AND_SPLIT인 경우는 프로세스 에이전트의 복제(Clone)를 통해 이후 참가자들에게 동일한 내용의 프로세스 에이전트가 파견되도록 한다. 전이 조건이 존재할 경우는 평가 결과가 참인 참가자에게만 파견된다.

③ XOR_SPLIT

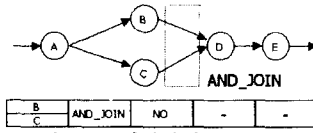


A	-	NO	XOR_SPLIT	전이 조건	C
				전이 조건	D

그림 5. B 참가자의 프로세스 테이블

XOR_SPLIT인 경우는 기술된 두 참가자 중 한 참가자에게만 프로세스 에이전트가 파견된다. 여러 참가자 중 WPDL에 기술된 순서대로 평가가 이뤄진다. 전이 조건이 존재할 경우는 평가 결과가 참인 참가자에게로 프로세스 에이전트가 파견된다.

④ AND_JOIN



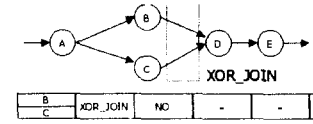
B C	AND_JOIN	NO	-	-	E

그림 6. D 참가자의 프로세스 테이블

AND_JOIN인 경우는 기술된 이전 참가자들로부터 동일한 프로세스 ID를 가진 프로세스 에이전트들이 모두 도착한 다음에 단위 작업이 수행된다.

⑤ XOR_JOIN

XOR_JOIN인 경우는 기술된 이전 참가자들로부터 각각 프로세스 에이전트가 도착한 경우 바로 단위 작업이 수행된다.



B C	XOR_JOIN	NO	-	-	E

그림 7. D 참가자의 프로세스 테이블

3.4 정의에 따른 프로세스 테이블의 간단한 예

그림 8에서 상단의 다이어그램은 프로세스 분기 및 전이 정보를 나타낸 것이다. 각 행은 주어진 프로세스 정의에 따라 각 참가자들이 가져야 하는 프로세스 테이블의 내용이다.

3.5 에이전트 기반 워크플로우 시스템 비교

표 3은 기존의 에이전트 기반 워크플로우 시스템과 제안한 시스템과의 상대적인 비교를 나타낸다. 제안한 워크플로우 시스템에서는 확장 시 참가자 측 프로세스 테이블 재구성 부담이

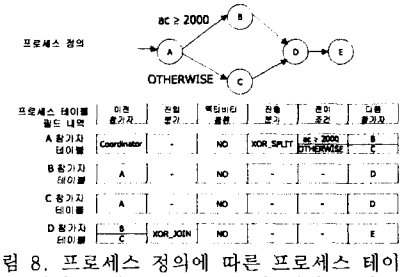


그림 8. 프로세스 정의에 따른 프로세스 테이블

존재한다. 그러나 스케쥴링 모듈을 각 참가자 별로 분산하여 수행 시 병목 발생 가능성을 줄이고, 에이전트 복제를 통한 병행 수행 지원, 이동 사용자 지원, 수행 중인 프로세스 정의 변경 시 최소 단위 작업 단위로 파급 효과를 줄일 수 있는 효과가 기대된다.

항목 / 종류	DartFlow[5]	스케쥴링 모듈을 이용한 워크플로우 시스템[6]	제안 워크플로우 시스템
스케쥴링 모듈의 집중/분산	분산 (프로세스 에이전트 단위)	집중 (워크플로우 엔진)	분산 (참가자 단위)
수행 중 병목 현상 발생 가능성	작업 리스트 서버에 병목 현상 발생	스케쥴링 모듈에 병목 현상 발생	없음
이동 에이전트 크기	크다 (산무형 모듈 포함)	작다	작다
이동 에이전트 수	작다	많다	작다
병행 수행 지원여부	지원 안함	지원	지원
수행 중인 프로세스 정의 변경	프로세스 에이전트 단위로 변경	서브 프로세스 단위로 변경	액티비티 단위로 변경
이동 사용자 지원	별도 이동한 지원	N/A	프로세스 에이전트 를 이동할 지원
이동 에이전트 수행 추적 지원	이비함 (작업 리스트 서버를 통한 간접적 지원)	계층적 관리를 통한 지원	별도의 관리 에이전트 이용
확장 시 참가자 측 부담	없음	없음	프로세스 테이블 구성 부담 존재

표 3. 이동 에이전트 기반 워크플로우 시스템 비교

4. 결론 및 향후 연구

기존의 워크플로우 시스템은 단위 업무간 제어 및 라우팅 등 스케쥴링 모듈이 워크플로우 서버에 존재하여 병목 현상으로 인한 성능 저하와 중앙 집중적 구조 특징으로 인해 확장성, 견고성, 가용성의 저하를 초래하였다. 본 논문에서 제안한 분산 스케쥴링을 지원하는 에이전트 기반 워크플로우 시스템에서는 각 참가자들은 자신이 수행해야 할 단위 작업을 알고 있으며 자신의 바로 전 참가자와 바로 다음 참가자를 알고 있다는 발상에서 출발한다. 즉 스케쥴링 정보와 스케쥴링 기능을 워크플로우 참가자 측에 일임하여 워크플로우 엔진의 부하를 줄여 성능을 향상시킬 수 있다. 또한 한 프로세스 인스턴스는 하나의 이동 에이전트에 의해 이뤄지므로 수행의 비동기적 특성 및 자율적 특성을 최대한 확보할 수 있다. 향후에는 제안한 워크플로우 시스템 구현과 프로세스 정의 변경 및 예외 처리 부분에 대한 연구가 필요하다.

5. 참고 문헌

[1] Workflow Management Coalition Specification, "Workflow Management Coalition Terminology & Glossary", WFMC-TC-1011, 1999.

[2] G. Alonso, D. Agrawal A. EL Abbadi, C. Mohan, "Functionality and Limitations of Current Workflow Management Systems", 1996.

[3] C. Mohan, "Recent Trends in Workflow Management Products, Standards and Research", 1997.

[4] Workflow Management Coalition Specification, "Workflow Management Coalition, The Workflow Reference Model", WFMC-TC-1003, 1995.

[5] Ting Cai, Peter A. Gloor, Saurab Nog, "DartFlow: A Workflow Management System on the Web using Transportable Agents", Technical Report PCS-TR96-283, 1996.

[6] 장재준, 정민아, 이도헌, "이동에이전트를 기반으로 한 워크플로우 스케쥴링 모듈의 설계", 한국정보처리학회 추계 학술발표논문집 제 6권 제 22호, 1999.