워크케이스 기반 워크플로우 엔진의 초대형성 성능 평가

Scalability Estimations of a Workcase-based Workflow Engine

안 형 진* 박 민 재** 이 기 원*** 김 광 훈****
Hyung-Jin Ahn Min-Jae Park Ki-Won Lee Kwang-Hoon Kim

요 약

최근 기업 및 조직들은 대규모 엔터프라이즈 환경에서 발생하는 대량의 업무 인스턴스들을 안정적으로 처리해줄 수 있는 초대형 워크플로우 관리 시스템 도입에 대한 활성화를 요구하고 있다. 워크플로우 벤더들은 초대형 워크플로우 서비스를 제공하는데 적합한 워크플로우 엔진을 구현하기 위해, 하드웨어들의 추가적인 연계를 통한 워크플로우 엔진의 물리적 성능 확장에 초점을 맞추고 있다. 그러나 워크플로우 엔진의 소프트웨어적인 아키텍처를 고려하지 않은 단순한 물리적 성능 확장은 불필요한 서비스 구축 시간 및 비용의 낭비를 야기할 수 있다. 이러한 한계를 개선하기 위하여, 본 논문에서는 워크플로우 엔진을 구성하는 소프트웨어 아키텍처를 고려하는 논리적 관점에서의 성능 향상을 위한 접근 방법을 모색한다. 이를 위해 워크플로우 서비스의 하드웨어 인프라를 전형적인 단위 클라이언트-서버 구조를 채택하고 있다고 가정하고, 대다수의 워크플로우 벤더들이 채택하고 있는 액티비티 인스턴스 아키텍처 기반 워크플로우 엔진과 본 논문에서 제안하는 워크케이스 아키텍처 기반 워크플로우 엔진의 성능을 비교 측정한다. 우리는 성능 측정의 분석 결과를 통해 논리적인 소프트웨어 아키텍처가 워크플로우 엔진의 초대형성에 많은 영향을 미칠 수 있음을 보여주고자 한다.

Abstract

Recently, many organizations such as companies or institutions have demanded induction of very large-scale workflow management system in order to process a large number of business-instances. Workflow-related vendors have focused on physical extension of workflow engines based on device-level clustering, so as to provide very large-scale workflow services. Performance improvement of workflow engine by simple physical-connection among computer systems which don't consider logical-level software architecture lead to wastes of time and cost for construction of very large-scale workflow service environment. In this paper, we propose methodology for performance improvement based on logical software architectures of workflow engine. We also evaluate scalable performance between workflow engines using the activity instance based architecture and workcase based architecture, our proposed architecture. Through analysis of this test's result, we can observe that software architectures to be applied on a workflow engine have an effect on scalable performance.

☞ Keywords : 초대형 워크플로우(Very Large-Scale Workflow), 워크케이스(Workcase) 기반 워크플로우 아키텍처 (Workflow-based Workflow Architecture), 워크플로우 초대형성 성능 평가(Workflow Scalability Estimations)

1. 서 론

기존의 기업들은 일반적으로 단위 조직의 인트라 환경 기반의 중·소규모 구성원들이 참여하는 비즈니스 프로세스의 자동화를 제공하는데 워크플로우 관리 시스템을 이용해왔다. 그러나 오늘날의 기업들은 비즈니스 리소스들의 확충 및 타 기업들과의 협력 관계 구축을 통한 조직 규모의 확장을 바탕으로 비즈니스 프로세스의 규모를 증대시켜 나가고 있다[1]. 초대형 워크플로우 관리 기술[6,7]은 대규모 엔터프라이즈 환경 상에서 발생하는 대량의 비즈니스 프로세스 관련 인스턴스들에 대한 빠르고 안정적인 처리를 제공하는 기술로써 현재 워크플로

* 정 회 원 : 경기대학교 컴퓨터과학과 박사과정
hjahn@kgu.ac.kr
** 정 회 원 : 경기대학교 컴퓨터과학과 박사과정
mean222@kgu.ac.kr
*** 준 회 원 : 핸디소프트 연구개발본부 사원
kwlee@handy.co.kr
**** 종신회원 : 경기대학교 컴퓨터과학과 교수
kwang@kgu.ac.kr
[2008/07/07 투고 - 2008/07/09 심사 - 2008/08/21 심사완료]

우 관리 기술 분야의 핵심적인 연구 이슈이다. 초대형 워크플로우 관리 기술을 구현하는 워크플로우 엔진은 다음과 같은 특징들을 충족하는 서비스를 제공해야 한다. 첫째, 초대형 워크플로우 엔진은 다수의 워크플로우 참여자들의 개입에 의한 이벤트 트리거링을 통해 발생되는 대량의 워크플로우 인스턴스들의 정상적인 구동을 지원하기 위한 시스템 리소스 공간의 유연한 확장성(Scalability)을 제공해야 한다. 둘째, 초대형 워크플로우 엔진은 해당 시스템 내에 거주하는 대량의 워크플로우 인스턴스들에 대하여 빠르면서도 안정적인 처리를 지원하는 신뢰성(Reliability)을 제공해야 한다. 셋째, 초대형 워크플로우 엔진은 대량의 워크플로우 인스턴스들을 처리하는 도중에 발생 가능한 예외 상황들에 대한 복구 및 백업을 통해 워크플로우 문맥적인 영속성을 보장하는 가용성(Availability)을 제공해야 한다.

최근 워크플로우 벤더들은 제품을 이용하는 조직들이 상기 기술된 특징들을 반영하는 초대형 워크플로우 서비스를 제공하도록 지원하기 위하여, 워크플로우 엔진을 구성하는 물리적인 시스템 장치들의 추가적 연결을 통해 성능 향상을 도모하는 하드웨어적 관점에서의 초대형 워크플로우 엔진 구현을 지향하고 있다. 그러나 단순한 하드웨어 추가에 의해 성능 향상을 추구하는 현재의 초대형 워크플로우 엔진 구축 방법은 물리적인 확장 구조에 적합한 소프트웨어 아키텍처의 반영을 고려하고 있지 않기 때문에, 추가된 하드웨어에 대비한 성능을 보장받기 어려우며 결과적으로 시스템 환경을 구축하는데 소요되는 시간 및 비용적 측면에서의 낭비를 초래할 수 있다.

본 논문은 기존 대다수의 워크플로우 엔진이 기반하고 있는 소프트웨어 아키텍처의 개선을 통해, 논리적 관점에서의 성능 향상을 도모하는 초대형 워크플로우 엔진 구축 방안을 모색하고자 한다. 이를 위해 본 논문에서는 워크플로우 엔진들의 하드웨어 인프라가 전형적인 단위 클라이언트-서버 환경이라는 전제 하에, 현재 대다수의 워크플로우 벤더들이 채택하고 있는 사실 표준인 OMG(Object Management Group) JointFlow[4]가 지향하는 액티비티 인스턴스 기반 아키텍처를 구현한 워크플로우 엔진과 본 논문에서 제안하는 워크케이스 아키텍처 기반 워크플로우 엔진의 성능을 비교 측정한다. 성능 측정에 관한 분석 결과를 토대로 워크플로우 엔진을 구성하는 소프트웨어 아키텍처가 초대형 워크플로우 서비스 환경을 구축하는데 매우 큰 영향을 미칠 수 있음을 보여주고자 한다.

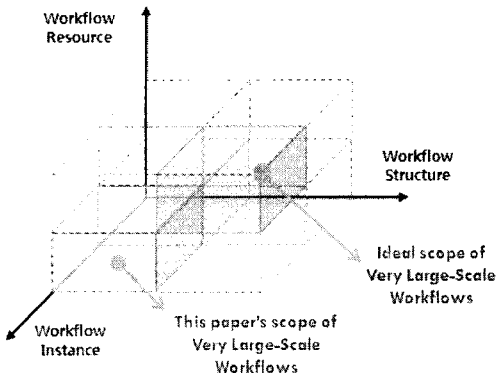
2. 관련 연구

초기의 OfficeTalk-D[5]와 같은 워크플로우 엔진들은 임의의 비즈니스 목표를 달성하는데 필요한 업무들의 순차적, 병행적 또는 선택적 절차들에 대한 자동화를 제공함으로써, 조직에서 일어나는 다양한 워크플로우 프로세스들의 처리에 소요되는 시간의 단축 및 그로 인한 비용 절감 효과를 실현해 주었다. 이러한 워크플로우 관리 시스템을 이용해 오던 조직들은 점차적으로 단순한 문서 및 작업에 필요한 파일들의 자동 전이가 아닌, 높은 복잡도를 가지는 비즈니스 프로세스들의 정확한 처리를 지원하는 워크플로우 엔진을 요구하게 되었다.

트랜잭션 지향적 워크플로우 엔진(Transactional Workflow Engine)[6]은 하나의 워크플로우 인스턴스 수행에 관한 전체 서비스 트랜잭션의 무결성을 보장하여 비즈니스 프로세스 자동화의 오류를 최소화하는 것을 목적으로 하는 시스템이다. Stuart et al.[7]은 CORBA를 이용하여 워크플로우 서비스 제공 시 발생하는 동적 예외 상황들에 대한 방지 및 복구를 지원하는 트랜잭션 워크플로우 엔진을 제안하였다. Choi et al.[8]은 다양한 플랫폼 운영 환경 상에서 트랜잭션 워크플로우 서비스 제공이 가능하도록 하기 위하여, 페트리-넷(Petri-net)을 이용한 워크플로우 모델링 방법을 제안하였다. 또한 K. Guntzel[9]는 웹 서비스 환경 상에서 동작하는 워크플로우 프로세스 인스턴스의 트랜잭션 무결성을 보장하기 위한 프로세스 모델링 방법 및 아키텍처를

제안하였다.

그러나 최근에 들어 여러 조직 구성원들이 참여하는 워크플로우 협업[10] 또는 조직들 간 워크플로우 협력[11,12] 등이 빈번해짐에 따라, 다수의 업무 수행자 또는 리소스들이 개입하여 발생되는 대량의 작업들을 안정적으로 처리해낼 수 있는 초대형 워크플로우 엔진(Very Large-Scale Workflow Engine)의 도입이 이슈화되고 있다. 임의의 워크플로우 엔진이 거대량의 워크플로우 인스턴스들을 원활하게 제어 및 관리하는 것이 가능하기 위해서는 해당 런타임 인스턴스들을 충분하게 용적할 수 있는 확장성이 우선적으로 보장되어야 하며, 이상적인 워크플로우 확장성을 보장하는 초대형 워크플로우 서비스를 제공하기 위해서는 그림 1에서 보는 바와 같이 세 가지 관점에서의 초대형성을 모두 고려할 수 있어야 한다[13].



(그림 1) 워크플로우 초대형성 구성 요소 및 관계

- 워크플로우 리소스 관점

하나의 워크플로우 리소스가 자신의 한 워크아이템을 처리하기 위해 워크플로우 엔진에게 이벤트 메시지들을 전송하는 시점에, 워크플로우 엔진은 해당 워크아이템의 처리를 위해 워크플로우 인스턴스를 생성 또는 활성화해야 한다. 이러한 사실로부터 워크플로우 리소스들이 프로세스 업무 처리에 개입할 시 워크플로우 엔진 내에는 그에 따른 처리를 위한 워크플로우 관련 인스턴스들이 발생한다는 점을 알 수 있으며, 결과적으로 워크플로우 리소스가 워크플로우 엔진의 확장성에 직접적으로 관여한다는 점을 알 수 있다.

- 워크플로우 프로세스 구조적 관점

두 기업체 이상이 참여하는 다자간 협력 관계에 의해 형성된 비즈니스 프로세스들은 일반적으로 참여 기업들이 관리하는 여러 비즈니스 프로세스들의 결합에 의해 다수의 액티비티들이 존재하는 구조를 가진다. 워크플로우 엔진이 높은 복잡도를 가지는 대규모 워크플로우 프로세스를 처리하기 위해서는 해당 워크플로우 프로세스로부터 발생되는 대량의 인스턴스들을 감내할 수 있는 확장성이 필수적으로 보장되어야 한다.

- 워크플로우 인스턴스 관점

초대형 워크플로우 서비스는 일반적으로 단위 워크플로우 엔진 또는 하나 이상의 워크플로우 엔진들로 구성된 워크플로우 수행 서비스(Workflow enactment service)들에 의해 최소 수천 개에서 최대 수백만 개에 이르는 인스턴스들에 대한 제어 및 관리가 가능한 서비스를 의미한다. 단위 워크플로우 엔진 또는 워크플로우 수행 서비스가 초대형 워크플로우 서비스 도메인 상에 상주하는 거대량의 워크플로우 인스턴스들을 빠르고 안정적으로 처리할 수 있기 위해서는 시스템의 성능을 저해하지 않음과 동시에 현재의 활성 인스턴스 양에 적절한 가용 공간을 유연하게 확장하는 것이 가능해야 한다.

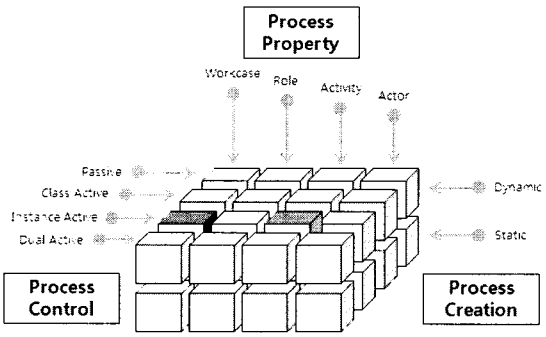
기존 대다수의 벤더들이 제공하고 있는 워크플로우 엔진들은 워크플로우 인스턴스 처리 가용 공간에 대한 확장성 확보를 위해서 하드웨어 중심적인 시스템 성능 증가를 추구해왔다. 워크플로우 벤더들은 하드웨어의 추가 증설을 통한 워크플로우 엔진 성능 확장 시에 물리적 관점에 치중한 시스템 컴포넌트 배치 방안들을 활용하고 있으나, 워크플로우 프로세스가 가진 특징들을 세부적으로 고려한

이상적인 배치라 하기 어렵다.

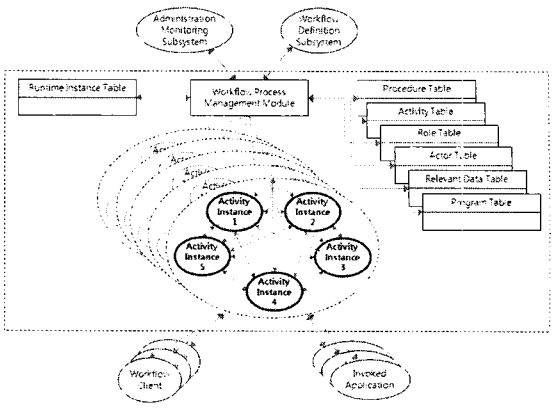
3. 워크케이스 기반 워크플로우 아키텍처

C. A. Ellis et al.[6]는 워크플로우 프로세스를 구성하는 업무 절차적 구조, 조직 구조 및 프로세스를 수행 방법 등의 다양한 관점들을 고려한 워크플로우 아키텍처의 개념적 분류 프레임워크를 제안하였다. 본 논문에서는 워크플로우 프로세스의 전반적인 수행을 능동적으로 관리하는 워크케이스라는 리소스 개념을 추가하여 기존의 분류 프레임워크를 확장하고 있다. 워크플로우 엔진을 구성하는 소프트웨어 아키텍처들은 워크플로우 프로세스의 수행을 주관하는 롤, 액터, 액티비티, 워크케이스 등의 리소스들의 관점을 고려하는 프로세스 속성(Process property) 항목과 워크플로우 엔진 상에서 워크플로우 프로세스의 컨텍스트를 처리하는 인스턴스 구조를 고려하는 프로세스 제어(Process control)항목 및 워크플로우 인스턴스 생성 방법에 관한 프로세스 생성(Process creation) 항목들을 통해 개념적인 분류가 가능하다.

현재 대부분의 워크플로우 엔진들은 워크플로우 표준화 그룹인 WfMC(Workflow Management Coalition)와 객체 관리 기술 표준화 그룹인 OMG에서 공동으로 제안하고 있는 표준 워크플로우 아키텍처인 JointFlow를 이용하고 있다. 그림 2에서 보는 바와 같이, OMG의 JointFlow는 액티비티 인스턴스들이 사전 정의된 워크플로우 프로세스의 절차에 따라 주도적으로 업무를 진행해나가는 구조를 가진 동적 인스턴스 액티브 유형의 액티비티 인스턴스 기반 워크플로우 아키텍처이다. 본 논문에서 제안하는 워크플로우 아키텍처는 워크플로우 엔진에 의해 생성 후 구동되는 프로세스 인스턴스인 워크케이스가 업무 절차에 따라 동적으로 액티비티 데이터들을 참조하여 워크플로우 프로시저를 처리해나가는 동적 인스턴스 액티브 유형의 워크케이스 기반 워크플로우 아키텍처이다.

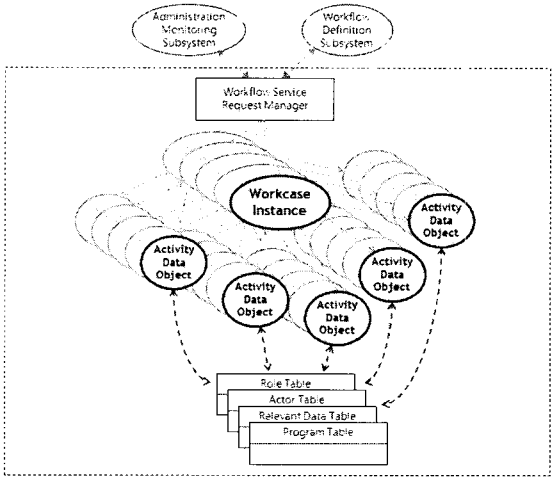


(그림 2) 워크플로우 아키텍처 개념적 분류 프레임워크



(그림 3) 액티비티 인스턴스 기반 워크플로우 아키텍처

m개의 액티비티들로 구성된 워크플로우 프로세

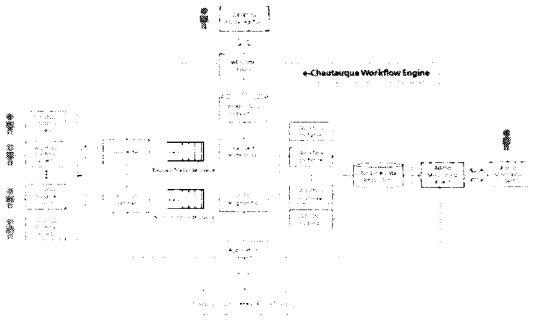


(그림 4) 동적 인스턴스 액티브 유형의 워크케이스 기반 워크플로우 아키텍처

스 p에 대하여 임의의 특정 워크플로우 리소스들의 개입에 의해 n번의 워크케이스 생성 요청 이벤트가 발생한다고 가정할 때, 그림 3과 4에서 보는 바와 같이 OMG의 JointFlow와 같은 액티비티 인스턴스 기반 워크플로우 아키텍처를 사용하는 워크플로우 엔진은 n * m개의 워크플로우 인스턴스들을 처리해야 하는데 반해, 워크케이스 아키텍처 기반의 워크플로우 엔진은 n개의 워크케이스 인스턴스들을 처리한다. 이로부터 워크케이스 기반 워크플로우 아키텍처는 액티비티 인스턴스 기반 워크플로우 아키텍처에 비해 워크플로우 인스턴스들의 처리량을 감소시키는 효과를 제공함으로서 워크플로우 엔진의 확장성을 증가시킨다는 것을 알 수 있다. 본 논문에서는 일반적인 서버-아키텍처 구조의 동일한 하드웨어 인프라 환경 상에서 워크케이스 기반 워크플로우 엔진이 기존의 일반적인 액티비티 인스턴스 기반 워크플로우 엔진보다 초대형 워크플로우 서비스를 제공하는데 적합한 구조임을 실제 성능 측정 실험을 통해 입증하고자 한다.

4. 워크플로우 엔진 설계 및 구현

이 장에서는 엔터프라이즈 도메인에서 일반적으로 이용하고 있는 액티비티 인스턴스 기반 워크플로우 아키텍처와 본 논문의 제안 아키텍처인 워크케이스 기반 워크플로우 아키텍처 간의 확장성에 관한 성능측정 결과를 비교 분석하기 위하여 e-Chautauqua라 명명하는 워크플로우 엔진을 설계 및 구현한다. e-Chautauqua 워크플로우 엔진은 액티비티 인스턴스가 프로세스 정의 템플릿의 정보에 의거하여 다양한 수행 담당 리소스들과의 상호 작용 및 업무 처리를 주관하는 전형적인 액티비티 인스턴스 기반 워크플로우 서비스 운영과 워크플로우 프로세스 인스턴스가 프로시저에 따른 적절한 업무 처리 시점에 액티비티들을 동적으로 데이터로써 활용하는 워크케이스 기반 워크플로우 서비스 운영 형태들을 모두 제공한다. 본 논문에서의 e-Chautauqua 워크플로우 엔진은 단위 서버-클라이



(그림 5) e-Chautauqua 워크플로우 엔진 시스템 아키텍처

언트, 클러스터링 기반 서버-클라이언트 또는 분산 환경 상에서 다수의 클라이언트들이 발생시키는 요청들에 대한 빠르고 안정적이며 지연 없는 처리를 지원하는 대표적인 서버-사이드 기술 중 하나인 Java EE(Java Platform, Enterprise Edition) EJB 프레임워크를 기반 기술로써 사용하고 있다. 그림 5에서 보는 바와 같이, e-Chautauqua 워크플로우 엔진의 주요 구성 컴포넌트들인 Workflow Client, Requester, Worklist Handler, Workflow Instance들은 하나의 워크플로우 프로세스에 대한 기본 서비스 트랜잭션을 시작하고 종료하기 위하여 다음과 같이 각자의 역할들을 수행한다.

Requester 컴포넌트는 기본적으로 액티비티 인스턴스들이 워크플로우 프로시저의 처리를 주도하도록 하는 구동 유형과 워크케이스가 프로세스의 전체적인 흐름을 주관하도록 하는 구동 유형의 두 가지 워크플로우 서비스 시작 오퍼레이션들을 제공한다. 워크플로우 프로세스의 서비스 시작 권한을 가지는 임의의 워크플로우 클라이언트는 Requester에서 제공하는 두 가지 시작 오퍼레이션들 중 하나를 이용하여 워크플로우 서비스 운영 방식을 선택하는 것이 가능하다. Requester로부터 생성된 워크케이스 또는 액티비티 인스턴스들은 실제 업무 처리를 위한 전처리로써 Worklist Handler를 통해 현 시점에서 처리되어야 하는 액티비티에 대한 워크아이템을 생성하여 해당 수행자 리소스들에게 할당하는 역할을 수행한다. 이 때, 액티비티 인스턴스 기반 워크

플로우 구조를 반영하는 서비스 운영 상에서는 액티비티 인스턴스가 Worklist Handler와의 커뮤니케이션을 통해 워크아이템 생성을 주관하며, 워크케이스 구조 기반의 운영 방식에서는 워크케이스가 현 시점에서의 액티비티 데이터에 상응하는 워크아이템을 생성시키는 책임을 가진다. 워크플로우 클라이언트는 제한된 시간 내에 자신에게 할당된 워크아이템을 처리한 후, Worklist Handler에게 업무 처리 종료 메시지를 전송한다. 워크아이템 종료 메시지를 수신한 Worklist Handler는 워크케이스 또는 대응 액티비티 인스턴스에게 해당 워크아이템이 종료되었음을 통보하고 다음 순서의 액티비티 처리를 진행해나가게 된다.

5. 성능 측정 및 결과 분석

본 장에서 우리는 초대형 워크플로우 서비스 제공에 적합한 워크플로우 아키텍처를 평가하기 위하여 두 가지의 성능 측정 기준 항목들을 설정한다. 첫 번째 기준 항목인 워크플로우 인스턴스 개수는 워크플로우 엔진이 워크플로우 클라이언트들의 개입 및 서비스 요청에 의해 발생되는 대량의 인스턴스들을 효율적으로 관리할 수 있는가를 판단하는 요소이다. 워크플로우 인스턴스 수의 한계는 클라이언트들이 요청하는 작업에 대한 정상적인 처리 및 관리가 가능한 최대 워크플로우 인스턴스 개수를 의미한다. 따라서 워크플로우 엔진 상에 상주하고 있는 대량의 워크플로우 인스턴스들에 관한 처리는 초대형 워크플로우 서비스 제공 가능 여부를 판단하는 지표로 사용할 수 있다. 두 번째 기준 항목인 평균 반응 시간은 워크플로우 클라이언트들의 작업 요청에 대한 워크플로우 인스턴스를 생성 및 구동하는데 소요되는 평균 시간을 의미한다. 워크플로우 클라이언트들로부터 전달되는 작업 요청의 증가에 따라 평균 반응 시간을 확인하고, 워크플로우 엔진이 관리하고 있는 시스템 자원의 한계로 인해 발생되는 병목 현상이 일어나는 부분을 확인하여 성능을 확인한다. 워크플로우 엔진의 평균 반응 시간의 연산 결과는 다음의 두 식들을 이용하여 획득한다.

$$W_{RT} = N_T + N_D + E_T + C_T + C_D + R_T + R_D$$

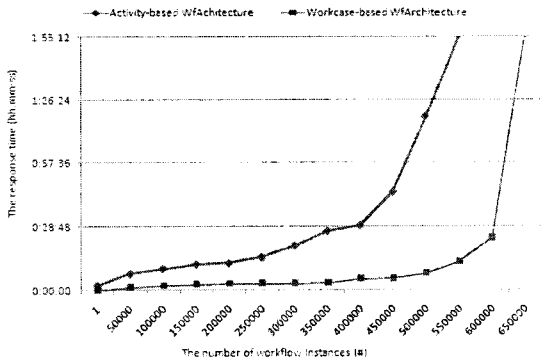
(식 1)

식 1에서 보는 바와 같이, 워크플로우 엔진이 하나의 워크플로우 인스턴스를 구동시키는데 소요되는 단위 워크플로우 인스턴스 반응 시간 W_{RT}는 한 워크플로우 클라이언트들의 작업 요청에 대한 이벤트 메시지를 워크플로우 엔진에게 전송하는데 소요되는 네트워크 트래픽 전송 소요시간 N_T, 클라이언트로부터 워크플로우 엔진으로의 요청 이벤트 메시지 전달 과정에서 발생되는 네트워크 트래픽 전송 지연 시간 N_D, 워크플로우 엔진이 클라이언트로부터 수신한 작업 요청 이벤트 메시지를 해석하여 인스턴스를 트리거링하는데 소요되는 시간 E_T, 이벤트 트리거링을 통해 워크플로우 인스턴스가 생성되는데 소요되는 시간 C_T, 클라이언트 작업 요청에 대응하는 워크플로우 인스턴스 생성 시 발생 가능한 시스템 지연 시간 C_D, 워크플로우 인스턴스가 해당 작업을 처리하기 위해 구동되는데 소요되는 시간 R_T와 워크플로우 인스턴스 구동 시 발생 가능한 시스템 지연 시간 R_D와 같은 요소들의 합이다.

$$R(n) = \frac{\sum_{i=1}^{n} W_{RT_i}}{n} \quad (n > 0)$$

(식 2)

식 2는 워크플로우 엔진이 관리하는 전체 워크플로우 인스턴스들에 대한 평균 반응 시간을 구하는 식이다. 평균 반응 시간 R은 워크플로우 엔진에 상주하는 전체 워크플로우 인스턴스 개수 n개의 각 반응 시간 W_{RT}들에 대한 평균 결과 값을 의미한다. 본 논문에서는 액티비티 인스턴스 기반



(그림 6) 성능 실험 대상 엔진들 간 초대형성 성능 측정 결과

아키텍처와 워크케이스 기반 아키텍처를 워크플로우 초대형성 성능 평가의 대상으로 선정하고, 두 워크플로우 아키텍처들이 단위 클라이언트-서버 구조의 하드웨어 인프라 상에서 동작하는 워크플로우 관엔진에 배치되어 동일한 워크플로우 프로세스 정의를 바탕으로 워크플로우 인스턴스를 발생시킨다고 가정한다. 또한 본 실험은 각 워크플로우 아키텍처의 인스턴스 관리 메커니즘에 의거하여 발생되는 워크플로우 인스턴스 개수와 각 인스턴스들이 생성 및 구동하는데 소요되는 반응 시간 등을 이용하여 성능 측정 결과를 비교 분석한다. 다음의 그림 6은 동일한 제약 조건상에서의 액티비티 인스턴스 기반 아키텍처와 워크케이스 기반 아키텍처 간 워크플로우 초대형성에 관한 성능 측정 결과를 나타낸 그래프이다.

워크플로우 초대형성을 평가하는 첫 번째 기준 항목인 워크플로우 인스턴스 수의 결과에서는 e-Chautauqua 워크플로우 엔진이 액티비티 인스턴스 기반 워크플로우 아키텍처를 선택하여 동작한 경우에는 약 42만 개에서 한계를 나타내고 있으며, 반면 본 논문에서 제안하고 있는 워크케이스 기반 워크플로우 아키텍처를 선택하여 동작한 경우에는 약 61만 개에서 한계를 보이고 있다. 이러한 결과로부터 알 수 있듯이, 액티비티 인스턴스 기반 워크플로우 아키텍처는 워크케이스 기반 워크플로우 아키텍처에 비하여 제어 관리해야 하는 대상 워크플로우 인스턴스 수가 보다 많음으로 인해 시스템의 병목 현상을 조기에 발생시키고 있음을 알 수 있다. 성능 평가의 다른 항목인 반응 시간의 결과에서는 e-Chautauqua 워크플로우 엔진이 워크케이스 기반 아키텍처를 선택한 경우에 더 낮은 반응 시간을 나타내고 있음을 확인할 수 있다. 또한 각 아키텍처들의 성능 측정 결과에서 병목 현상이 발생되는 시점의 반응 시간들을 제외한 평균 반응 시간에 대한 결과는 액티비티 인스턴스 기반 아키텍처를 선택한 경우에는 31분 3초인데 반해, 워크케이스 기반 아키텍처를 이용하여 동작한 경우에는 5분 31초라는 상당히 큰 차이의 결과를 나타내고 있다. 이로부터 우리는 워크케이스 아키텍처 기반 워크플로우 엔진이 액티비티 인스턴스 기반 엔진에 비해 대량의 워크플로우 인스턴스들을 보다 신뢰적이고 빠르게 처리한다는 것을 알 수 있다.

6. 결 론

본 논문에서는 워크플로우 엔진의 소프트웨어적 관점에서 초대형 워크플로우 서비스 제공이 가능한 환경 구축을 지원하는 워크케이스 기반 워크플로우 아키텍처를 제안하고 있다. 워크케이스 기반 워크플로우 아키텍처는 OMG JointFlow 아키텍처로 대표되는 기존의 액티비티 인스턴스 기반 워크플로우 아키텍처에 비해 워크플로우 엔진이 제어 관리해야 하는 인스턴스들의 수를 대폭적으로 감소시킴으로써 시스템의 확장성을 증대시키기 위한 아키텍처이다. 이를 증명하기 위해, 본 논문에서는 현재 대부분의 워크플로우 엔진들이 이용하고 있는 액티비티 인스턴스 기반 워크플로우 아키텍처와 본 논문의 제안 아키텍처인 워크케이스 기반 워크플로우 아키텍처를 동일한 하드웨어 인프라 상에서 동작시켜 두 아키텍처들의 워크플로우 초대형성을 실험하였다. 본 논문의 성능 측정 결과로부터 알 수 있듯이, 워크케이스 기반 워크플로우 아키텍처를 이용한 워크플로우 엔진이 액티비티 인스턴스 기반 아키텍처에 비하여 관리 가능한 워크플로우 인스턴스 수의

한계치가 더 높으며, 각 워크플로우 인스턴스들을 생성 및 구동하는데 소요되는 평균 반응 시간이 더 낮음으로써 보다 나은 초대형성 성능을 보인다는 것을 확인할 수 있었다.

향후에는 워크케이스 기반 워크플로우 아키텍처가 분산 클러스터링 및 그리드/P2P 인프라에서도 기존 아키텍처에 비해 워크플로우 초대형성을 보장하는 결과를 낼 수 있는지에 대해 실험을 통해 증명하고자 한다.

Acknowledgement

본 연구는 경기대학교 교내 연구그룹 과제 지원 사업(2006-020)의 연구 결과로 수행되었음.

참 고 문 헌

[1] C. A. Ellis, K. Kim, "Process Aware Information Systems: A Human Centered Perspective ", The Lecture Notes in Computer Science, Vol 4505, pp.39-49, 2007

[2] K. Kim et al., "An EJB-based Very Large-Scale Workflow Management System and Its Performance Measurement", The Proceedings of International Conference on Web-Aged Information Management, The Lecture Notes in Computer Science, Vol. 3739, pp.526-537, 2005

[3] J. Won, K. Kim, "Workcase-Oriented Workflow Enactment Components for Very Large Scale Workflows", The Lecture Notes in Computer Science, Vol. 4797, pp.910-919, 2007

[4] Object Management Group(OMG) and Workflow Management Coalition(WfMC), "Workflow Management Facility - JointFlow, v1.2", 2000

[5] C. A. Ellis, M. Bernal, "OfficeTalk-D: An Experimental Office Information System", The Proceedings of SIGOA Conference on Office Information Systems, pp.131-140, 1982

[6] C. A. Ellis, K. H. Kim, "A Framework and Taxonomy for Workflow Architectures", The Proceedings of ACM Group2000: The 4th International Conference on Design for Cooperative Systems, 2000

[7] S. M. Wheater, S. K. Shrivastava, F. Ranno, "OPENflow: A CORBA Based Transactional Workflow System", The Journal of the Advances in Distributed Systems, The Lecture Notes in Computer Science, Vol. 1752, pp.354-374, 2000

[8] I. Choi, C. Park, C. Lee, "Task-Net: Transactional Workflow Model based on Colored Petri Net", The European Journal of Operational Research, The ELSEVIER, Vol. 136, Issue 2, pp.383-402, 2002

[9] K. Guntzel, "Web Services-Based Transactional Workflows: Advanced Transaction Concepts", The Proceedings of the OTM Workshops 2003, The Lecture Notes in Computer Science, Vol. 2889, pp.70-82, 2003

[10] K. Kim, I. Ra, "e-Lollapalooza: A Process-Driven e-Business Service Integration System for e-Logistics Services", KSII Transactions on Internet and Information Systems, Issue 1, 2007

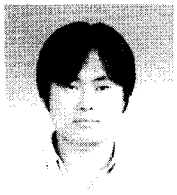
[11] K. Kim, "A Process-Driven Inter-organizational Choreography Modeling System", The Lecture Notes in Computer Science, Vol. 3762, pp.485-494, 2005

[12] J. Vonk, W. Derks, P. Grefen, M. Koetsier, "CrossFlow: Cross-Organizational Transaction Support for Virtual Enterprises", The Proceedings of the Cooperative Information Systems, Vol. 1, pp.323-334, 2006

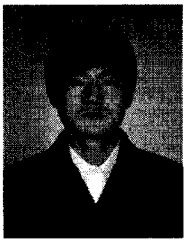
[13] R. A. Dheepak et al., "Scalable Enterprise Level Workflow Manager for the Grid", The Proceedings of the International Workshop on Grid and Peer-to-Peer based Workflows, 2005

◐ 저 자 소 개 ◑

안 형 진
2004년 경기대학교 전자계산학과 졸업(학사)
2006년 경기대학교 대학원 전자계산학과 졸업(석사)
2006년~현재 경기대학교 대학원 컴퓨터과학과 박사과정
관심분야 : 워크플로우/BPM 기술, RFID/USN, 엔터프라이즈 그리드 워크플로우 기술
E-mail : hjahn@kgu.ac.kr



박 민 재
2004년 경기대학교 전자계산학과 졸업(학사)
2006년 경기대학교 대학원 전자계산학과 졸업(석사)
2006년~현재 경기대학교 대학원 컴퓨터과학과 박사과정
관심분야 : 워크플로우/BPM 기술, 워크플로우 마이닝 기술
E-mail : mean222@kgu.ac.kr



이 기 원
2006년 경기대학교 전자계산학과 졸업(학사)
2008년 경기대학교 대학원 컴퓨터과학과 졸업(석사)
2008년~현재 핸디소프트 연구원
관심분야 : 워크플로우/BPM 기술, 서비스 지향 워크플로우 엔진 기술
E-mail : kwlee@handysoft.co.kr

김 광 훈
1984년 경기대학교 전자계산학과 졸업(이학사)
1986년 중앙대학교 대학원 전자계산학과 졸업(이학석사)
1994년 Univ. of Colorado at Boulder, 컴퓨터과학과 졸업(이학석사)
1998년 Univ. of Colorado at Boulder, 컴퓨터과학과 졸업(이학박사)
1986년~1991년 한국전자통신연구원 연구원
1998년~현재 : 경기대학교 정보과학부 컴퓨터과학과 교수
2006년~현재 : GRRC 콘텐츠 융합 소프트웨어 연구센터 센터장
관심분야 : 워크플로우/BPM 기술, RFID/USN, CSCW, 웹서비스, 분산처리기술, 데이터베이스
E-mail : kwang@kgu.ac.kr