시간 제약을 가진 워크플로우의 성능 향상 기법

(A Schema to Improve the Performance of Workflow with Time Constraints)

손 진 현 † 김 명 호 ††
(Jin Hyun Son) (Myoung Ho Kim)

요 약 시간 제약을 가지는 워크플로우에서 만약 워크플로우 인스턴스가 마감 시간 안에 종료되지 못하면, 워크플로우 관리 시스템은 보상 처리와 같은 예외적인 조치들을 취할 것이다. 이러한 상황은 시스템에게 과부하를 주기 때문에 워크플로우 처리의 성능을 저하시킨다. 이에 본 논문에서는 고성능 워크플로우 시스템을 지원하기 위해 마감 시간을 만족하는 워크플로우 인스턴스들의 수를 증가시킬 수 있는 새로운 기법을 제안한다. 이를 위해 먼저 워크플로우에서 가장 긴 평균 실행 시간을 요구하는 임계 경로를 찾는 방법을 제안한다. 그리고 임계 경로에 속하는 각 임계 액티비티의 처리 용량을 향상시키기 위해 최소 중복 서버의 개수를 결정하는 방법을 제안한다. 마지막으로, 실험들을 통하여 본 논문에서 제안된 기법의 유용함을 보인다.

Abstract If a workflow instance misses its given deadline in time-constrained workflows, workflow management systems may take exceptional behaviors such as compensating processing, which makes the system overloaded and degrades the performance of workflow processing. In this respect, the paper proposes a scheme to be able to increase the number of workflow instances satisfying the given deadline. We first present a method to find out the critical path which is the longest execution path in a workflow. And then, we develop a method to determine the minimum number of duplicated servers for the critical activity in order to give it sufficient processing capacity. Finally, we verify the usefulness of our scheme by the experiments.

1. 서 론

워크플로우는 자동화 및 전산화 된 업무 프로세스로서 제어 흐름에 따라 연결된 액티비티들로 구성된다. 각 액티비티는 자신에게 할당된 역할을 수행하는 사람 혹은 소프트웨어 프로그램과 같은 관련 서버들을 가진다. 워크플로우 관리 시스템은 액티비티들 사이의 제어 흐름을 관리하고 관련된 서버들을 구동함으로써 워크플로우의 자동화를 지원한다[1]. BPR, 미들웨어, 객체 지향 시스템 등과 같은 많은 관련 기술들의 출현은 워크플로우에 대한 관심을 더욱 고조시키고 있으며, 특히 이들 기술들은 실제 업무 환경에 적용될 수 있는 워크플로우 관리 시스템의 개발을 가능하게 한다.

일반적으로 워크플로우를 정의할 때 마감 시간과 같은 워크플로우의 시간 제약들을 지정할 수 있다[2][3]. 만약 워크플로우 인스턴스가 워크플로우 마감 시간 안에 종료되지 못하면, 워크플로우 관리 시스템은 이미 수행된 액티비티들에 대한 보상 처리(compensating processing)와 같은 예외적인 행동들을 취할 것이다. 일반적으로 워크플로우 환경에서 이러한 보상 처리는 사람이 개입되는 경우가 많이 있다. 워크플로우 마감 시간을 어긴다는 것은 위와 같은 보상 처리로 인한 부가적인 시스템 과부하와 시스템 자원을 낭비하게 되어 결국

· 본 연구는 한국과학재단 특정기초연구(과제번호 1999-1-303-007-3) 지원으로 수행되었음
† 비 회 원 : 한국과학기술원 전자전산학과
jhson@dbserver.kaist.ac.kr
†† 종신회원 : 한국과학기술원 전자전산학과 교수
mhkim@dbserver.kaist.ac.kr
논문접수 : 1999년 5월 24일
심사완료 : 2000년 1월 27일

워크플로우 시스템 성능의 저하를 가져온다. 그러므로 시간 제약을 가진 워크플로우 처리의 성능 개선을 위해서 마감 시간을 만족하는 워크플로우 인스턴스의 수를 가능한한 증가시킬 수 있는 기법의 개발이 요구된다.

워크플로우에는 여러 수행 경로들이 존재하는데 이들 중에서 평균 실행 시간이 가장 긴 하나의 순차 경로를 임계 경로(critical path)라고 한다. 워크플로우에서 각 액티비티는 자신의 역할을 수행하기 위한 시간이 필요하므로, 전체 워크플로우의 실행 시간은 임계 경로에 의해서 결정된다. 만약 임계 경로에 속하는 액티비티(이를 임계 액티비티라고 함)들의 실행 시간이 지연되면, 워크플로우의 전체 처리 시간이 직접적으로 영향을 받게 된다. 그러므로, 워크플로우 서비스 요청율에 대해서 가능한 한 많은 워크플로우 인스턴스들이 마감 시간 안에 수행되기 위해서는 임계 액티비티들에 대한 충분한 처리 용량이 제공되어야 한다. 한가지 방법으로 액티비티 서버의 중복은 이 목적을 달성할 수 있다. 그러나 여기서 주목해야 할 사항은 과도한 중복 서버들의 지원은 시스템 자원을 낭비하여 시간 제약을 가진 워크플로우의 성능을 개선하는데 도움을 주지 못한다는 것이다[4][5].

액티비티 마감 시간의 계산, 시간 제약 사항들에 대한 검사, 에스컬레이션 관리 등과 같은 워크플로우 관리 시스템에서 고려될 수 있는 시간 관리 문제들은 PERT를 확장하여 [6]에서 언급되었다. [6]에서의 액티비티 마감 시간은 워크플로우 설계 시간(workflow build-time)에 정적으로 설정되는 반면에, [3], [7], 그리고 [8]은 예측 워크플로우(predictive workflow)에서 액티비티에 대한 동적 마감 시간 할당과 에스컬레이션을 관리하는 방법에 대해서 연구하였다. 이러한 연구들은 모두 분산 소프트 실시간 환경에서 다양한 마감 시간 할당 방법들을 제안한 [9]에 기반을 두고 있다. 분산 소프트 실시간 시스템에서 하나의 작업(task)은 여러 개의 부작업(subtask)들로 구성되며 이들 부작업들은 수행 순서에 따라 서로 연결되어 있다. 이는 워크플로우 시스템에서 워크플로우와 그를 구성하는 액티비티들 사이의 관계에 대응된다. 그러므로 [9]에서 제안된 마감 시간 할당 방법들은 워크플로우 시스템에 쉽게 적용될 수 있다. [3]에서도 역시 [9]에서 제안된 방법들을 기반으로 하면서 새로운 동적 마감 시간 할당 방법들을 제안하였다.

기존의 시간 제약을 갖는 워크플로우 연구에서는 워크플로우 처리량(workflow throughput)을 개선하기 위해 각 액티비티의 마감 시간을 효율적으로 할당하는 방법들을 제안하였다. 그러나 이 방법들은 워크플로우의 사용자 수가 증가하여 워크플로우 시스템에 과부하가 발생하면 효과적으로 적용될 수 없는 한계를 가진다. 반면에, 본 논문에서는 시간 제약성 워크플로우에서 임계 액티비티들을 찾고 각 임계 액티비티의 처리 용량을 증가 시키기 위해 최소 중복 서버의 수를 결정하는 방법을 제안한다. 제안된 방법의 주요 공헌으로는 워크플로우의 임계 경로를 분석하여 임계 액티비티에 대응하는 서버의 개수를 결정하는 새로운 방법의 제안과 이를 바탕으로 워크플로우 마감 시간을 만족하는 워크플로우 인스턴스의 수를 증가시켜 고성능 워크플로우 시스템을 지원할 수 있다는 것이다.

본 논문은 다음과 같이 구성되어 있다. 2절은 본 논문에서 고려하는 워크플로우 구조에 대해 설명한다. 3절에서는 워크플로우의 임계 경로를 정의하고 이를 찾는 방법을 제시한다. 각 임계 액티비티에 대한 최소 중복 서버의 개수를 결정하는 방법은 4절에서 언급한다. 5절에서는 본 논문에서 제안된 방법들에 대해 분석하고 다양한 실험 결과들을 제시한다. 마지막으로, 6절에서는 본 논문의 결과 및 공헌, 그리고 앞으로 연구해야 할 사항들에 대해 요약함으로써 결론을 맺는다.

2. 워크플로우 처리 구조

워크플로우는 액티비티들이 순차, 병행(AND), 선택(OR), 반복(LOOP) 등의 네가지 제어 구조들에 의해서 상호 연결된 액티비티 네트워크이다[1]. 워크플로우 인스턴스는 실행되고 있는 실제 워크플로우 프로세스이다.

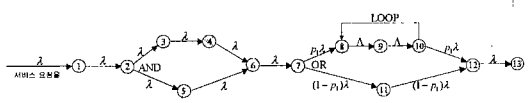


그림 1 워크플로우 대기 행렬 네트워크

본 논문에서 우리는 각 액티비티의 서버는 지수 서비스 시간 분포를 가지고, 사용자의 워크플로우 서비스 요청은 포아송 과정으로 도착한다고 가정한다. 이에 워크플로우는 기존의 전화 네트워크와 컴퓨터 통신 네트워크와 같이 M/M/1 대기 행렬 네트워크로 모델링 되고 워크플로우를 구성하는 각 액티비티는 하나의 독립적인 M/M/1 대기 행렬 시스템이 된다. 그러므로 그림 1과 같은 M/M/1 대기 행렬 네트워크인 워크플로우에서 사용자의 서비스 요청율, 각 액티비티 서버의 서비스율, 그리고 OR과 LOOP 제어 구조에서 분기 확률 등이 설

정될 때, 모든 액티비티들에서의 도착과 출발 과정을 명시할 수 있다. 이는 시간 역행성(time reversibility)과 아래에서 언급된 사실들을 통해서 가능하다[10].

그림 1에서 OR 제어 구조의 액티비티 7과 12처럼 독립적인 포아송 과정들의 분할과 병합은 역시 포아송 과정이 된다고 알려져 있다[10]. AND 분할 액티비티에서의 도착과 출발 과정 역시 포아송 과정이다. 그러나, AND 병합 액티비티(그림 1의 액티비티 6)의 서버가 구동되기 전에 그 액티비티로 전달되는 모든 서비스 요청들이 동기화 되어야 하기 때문에, AND 병합 액티비티의 도착 과정은 완전한 포아송 과정이 아니다. 그럼에도 불구하고 우리는 아래와 같은 이유로 AND 병합 액티비티에서의 도착 과정을 포아송 과정이라고 가정할 때 발생하는 에러를 무시할 수 있다. n개의 경로를 가진 AND 분할 액티비티에서 워크플로우 인스턴스는 동시에 수행되는 독립적인 n개의 하위 요청들로 분할 된다. 이들 하위 요청들은 모두 포아송 과정들이다. 그리고 n개의 하위 요청들은 AND 병합 액티비티에서 서로 동기화되어야 한다. 동기화 시점은 n개의 하위 요청들 중에서 AND 병합 액티비티에 가장 늦게 도착하는 요청에 의해서 결정된다. 그러므로, AND 병합 액티비티에서의 도착 과정은 AND 제어 구조의 n개의 경로들 중에서 가장 긴 실행 시간을 갖는 경로를 통해서 전달되는 하위 요청들에 의해서 결정되므로 거의 포아송 과정이라고 간주 할 수 있다. LOOP 제어 구조는 서비스 요청들이 이미 지나온 액티비티들로 돌아가는 피드백(feedback)을 가진다. 비록 LOOP의 내부 흐름은 이러한 피드백에 의해서 포아송 과정은 아니지만, LOOP 내부의 각 액티비티는 마치 독립적인 M/M/1 시스템인 것처럼 동작한다고 알려져 있다[11]. 그러므로 LOOP 제어 구조 역시 M/M/1 대기 행렬 네트워크의 부분이 될 수 있다. 본 논문에서 우리는 워크플로우의 AND, OR, LOOP 제어 구조들을 비순차 제어 구조라고 부르기로 한다.

워크플로우는 액티비티들 사이의 임의의 제어 흐름을 명시할 수 있도록 허용하기 때문에 세밀한 주의 없이 사용한다면 최종적으로 생성된 워크플로우를 이해하기 힘들뿐만 아니라 오류를 교정하기도 힘들다[12]. 그래서, [3]에서는 완전 구조적 워크플로우(well-structured workflow)의 개념을 이용하여 워크플로우들을 정의하고 있다. 본 논문에서는 [3]의 워크플로우 구조를 확장한 중첩 워크플로우(nested workflow)를 정의한다.

정의 2.1 워크플로우에서 비순차적 제어 구조가 임의의 다른 워크플로우 제어 구조들을 포함할 때, 이러한 워크플로우를 **중첩 워크플로우(nested workflow)**라고 부른다. 여기서, LOOP 제어 구조가 임의의 비순차적 제어 구조 C에 포함될 때, LOOP의 피드백 지점 역시 C안에 포함되어야 한다.

정의 2.2 내포형 워크플로우에서 가장 바깥쪽의 비순차적 제어 구조를 비순차적 제어 블록, 간단히 **제어 블록(control block)**이라고 부른다.

그림 1에는 AND 제어 블록과 LOOP를 포함하는 OR 제어 블록이 있다. 만약 LOOP의 피드백 지점이 액티비티 8 대신에 액티비티 4이면, 워크플로우는 의미있는 제어 흐름을 갖지 못한다. 그러므로 본 논문에서는 중첩 워크플로우 구조에 의해서 정의된 워크플로우들을 고려한다.

3. 임계 경로

워크플로우 시스템에는 동시에 많은 워크플로우 인스턴스들이 실행될 수 있기 때문에, 만약 한 액티비티의 서버가 어떤 워크플로우 인스턴스에게 할당되었다면 그 서버를 이용하려는 다른 워크플로우 인스턴스들은 기다려야 한다. 그러므로, 한 액티비티에서 워크플로우 인스턴스의 평균 실행 시간(average execution time)은 그 액티비티 서버의 평균 서비스 시간(average service time)과 그 액티비티의 대기 행렬에서 기다려야 하는 평균 대기 시간(average waiting time)의 합으로 계산된다[5] [10].

```
1. 제어 블록 안에 있지 않은 모든 순차 제어 구조들은
   임계 경로에 속한다.
2. WHILE (모든 제어 블록)
   {
     3. IF (제어 블록에서 가장 내부의 제어 구조가
            AND 혹은 OR)
            제어 구조에서 가장 긴 평균 실행 시간을
            가지는 경로를 선택.
     4. IF (제어 블록에서 가장 내부의 제어 구조가
            LOOP)
            LOOP를 순차 제어 구조로 변환.
     5. IF (제어 블록의 부임계 경로가 아직 결정되지
            않음)
            GO TO 3.
   }
6. 모든 부임계 경로들을 연결하여 워크플로우의 임계
   경로를 구성.
```

그림 2 ICSF 알고리즘

임계 경로는 한 워크플로우의 시작에서 종료까지 가장 긴 평균 실행 시간을 가지는 일련의 액티비티 순차 구조이다[8]. 임계 경로에 속하는 액티비티들을 임계 액티비티(critical activity)라고 부른다. 워크플로우의 네가지 제어 구조들은 워크플로우 처리 시간에 큰 영향을 주기 때문에, 임계 경로를 찾기 위해서는 반드시 이들

구조들의 특징들을 고려해야 한다. 그러므로, 본 논문에서는 임계 경로를 찾기 위해서 워크플로우 제어 구조들과 한 액티비티에서 워크플로우 인스턴스의 평균 실행 시간을 기반으로 하고 있다.

본 논문에서 제안된 ICSF(Innermost Control Structure First)는 시간 제약을 가지는 중첩 워크플로우에서 임계 경로를 찾는 방법이다. 그림 2는 ICSF의 알고리즘을 설명하고 있다. ICSF는 먼저 워크플로우의 각 제어 블록에서 가장 긴 평균 실행 시간을 갖는 부임계 경로(sub-critical path)를 결정한 후 이들 부임계 경로들을 서로 결합하여 전체 워크플로우의 임계 경로를 찾는다. 각 제어 블록에서 부임계 경로는 그 제어 블록의 가장 내부의 제어 구조에서부터 가장 바깥쪽에 있는 제어 구조의 방향으로 가장 긴 실행 경로를 점차적으로 선택함으로써 결정된다. 만약 가장 내부의 제어 구조가 AND 혹은 OR 제어 구조이면, 이들 구조에 속하는 여러 개의 경로들 중에서 가장 긴 평균 실행 시간을 갖는 경로를 선택한다. 만약 가장 내부의 제어 구조가 LOOP 제어 구조이면, ICSF는 LOOP 제어 구조를 분석하기 용이한 순차 제어 구조로 변환한다. 제어 블록들에 속하지 않는 모든 순차 제어 구조들은 워크플로우를 수행하는데 있어서 유일한 경로들이므로 모두 임계 경로에 속한다.

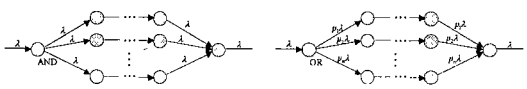


그림 3 AND 혹은 OR 제어 구조

M/M/1 대기 행렬 시스템에서 평균 실행 시간은 $W=\frac{1}{\mu-\lambda}$ (단, λ: 서비스 요청율, μ: 서버의 서비스율)이다. 순차 대기 행렬 네트워크인 순차 액티비티 경로의 평균 실행 시간은 각 액티비티의 평균 실행 시간의 합이다. 그러므로 그림 3과 같은 AND 혹은 OR 제어 구조에서 가장 긴 실행 경로(즉, 부임계 경로)는 아래와 같이 결정된다. 액티비티 i에서의 평균 실행 시간이 $W_i=\frac{1}{\mu_i-\lambda_i}$ 일 때,

부임계 경로 = 각 경로에 속하는 모든 액티비티들에 대해

$$MAX\left(\sum\frac{1}{\mu_i-\lambda_i}\right)$$ 인 순차 경로.

여기서 μ_i는 액티비티 i에 대한 서버의 서비스율이고, 액티비티 i에 대한 서비스 요청율 λ_i는 AND 제어 구조인 경우는 λ이며 분기 확률 p_i을 가진 OR 제어 구조인 경우는 $p_i\lambda$이다.

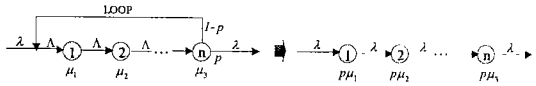


그림 4 LOOP 제어 구조

LOOP 제어 구조에서 각 액티비티는 2절에서 언급한 것처럼 하나의 독립적인 M/M/1 대기 행렬 시스템으로 간주될 수 있기 때문에, LOOP 제어 구조는 아래의 과정을 통해서 상응하는 순차 제어 구조로 변환된다. 그림 4에서 도착률 Λ는 LOOP의 피드백으로 인해서

$$\Lambda=\lambda+(1-p)\Lambda$$
$$\Lambda=\frac{\lambda}{p}$$

이다. 그러므로, LOOP 제어 구조의 내부는 도착률 Λ을 가지는 순차 제어 구조로 간주될 수 있다. 이를 도착률 λ인 순차제어 구조로 변환하는 과정은 아래와 같다. 액티비티 i의 서버 효율 ρ_i는

$$\rho_i=\frac{\lambda}{p\mu_i},\quad i=1,\ldots,n$$

여기서 μ_i는 액티비티 i에 대한 서버의 서비스율이다. 그러므로 LOOP 제어 구조의 평균 실행 시간은

$$E[R]=\left(\frac{\rho_1}{1-\rho_1}+\frac{\rho_2}{1-\rho_2}+\ldots+\frac{\rho_n}{1-\rho_n}\right)\frac{1}{\lambda}$$
$$=\frac{1}{p\mu_1-\lambda}+\frac{1}{p\mu_2-\lambda}+\ldots+\frac{1}{p\mu_n-\lambda} \quad (1)$$

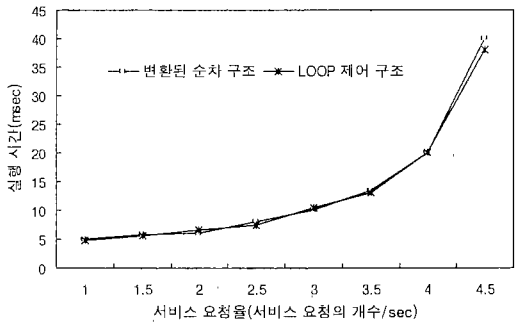


그림 5 LOOP 제어 구조의 변환

이다. $\frac{\rho_i}{1-\rho_i}$는 액티비티 i에서 머무르고 있는 서비스 요청들의 수이다. 식 (1)은 서비스율이 $p\mu_i$인 n개의 액티비티들로 구성된 순차 제어 구조의 평균 실행 시간과 동일하므로, LOOP 제어 구조를 그림 4와 같이 순차 제어 구조로 변환시킬 수 있다. 그림 5의 실험 결과는 이 변환 과정을 검증한다. 즉, LOOP 제어 구조와 변환된 순차 제어 구조의 시뮬레이션 결과가 거의 동일함을 알 수 있다.

본 논문에서 제안된 ICSF을 사용하여 워크플로우의 임계 경로를 찾는 한 예로써 그림 1은 두개의 제어 블록들을 가지고 있다. 하나는 AND 제어 블록이고, 다른 하나는 LOOP 제어 구조를 포함하는 OR 제어 블록이다. 이들 두 제어 블록들의 가장 내부의 제어 구조들은 각각 AND와 LOOP이다. 먼저 위에서 언급한 방법으로 AND 제어 블록에서 부임계 경로를 얻는다. 그리고 LOOP를 순차 구조로 변환한 후 OR 제어 블록에서 역시 부임계 경로를 얻는다. 이들 부임계 경로들을 연결함으로써 그림 1 워크플로우의 임계 경로를 찾을 수 있다.

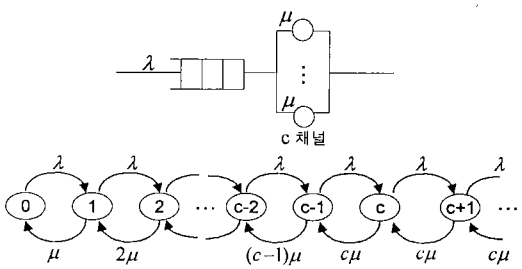


그림 6 M/M/c 모델

4. 최소 서버의 개수

임계 액티비티에 대한 서버의 처리 시간은 서버를 중복함으로써 개선될 수 있다. 이들 중복 서버들은 임계 액티비티에 대한 서버 클래스를 구성한다. 서버 클래스에 속하는 서버들은 모두 동일한 서비스를 제공하기 때문에, 서버 클래스는 그림 6과 같이 c개의 병렬 채널을 갖는 M/M/c로 모델링될 수 있다. 워크플로우 처리 과정에서 워크플로우 관리 시스템(WFMS)은 스케줄러의 역할을 담당한다. WFMS는 사용자의 워크플로우 서비스 요청에 대해 해당 워크플로우의 첫번째 액티비티의 서버를 구동시킨다. 구동된 서버는 자신의 역할을 수행한 후 그 결과를 WFMS에 알린다. 이후 WFMS는 첫번째 액티비티와 직접적인 제어 관계로 연결된 다음 액티비티들의 서버들을 구동시킨다. 이와 같은 WFMS와 액티비티들 사이의 제어 흐름은 전체 워크플로우가 종료될 때 까지 반복되며, WFMS는 액티비티들 사이에서 스케줄러의 역할을 담당한다. 그러므로 워크플로우에서 액티비티들 사이의 전이는 그림 7과 같은 3계층(3-tier) 클라이언트/서버 시스템으로 간주될 수 있다. 한 액티비티와 관련된 서버들은 서버 클래스를 구성하고, 그 액티비티와 직접적인 제어 흐름을 가지는 액티비티들의 서버들은 클라이언트 클러스터(client cluster)를 형성한다. WFMS는 클라이언트 클러스터로부터 서버 클래스로 서비스 요청들을 전달한다. 예를 들면, 그림 1에서 액티비티 10과 11에 대한 서버들은 액티비티 12의 클라이언트 클러스터를 구성한다.

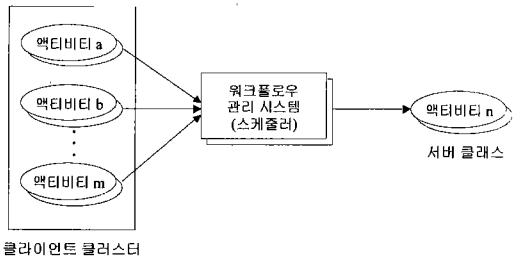


그림 7 스케줄러 역할을 담당하는 워크플로우 관리 시스템

3계층 클라이언트/서버 시스템에서 대부분의 실행 시간은 M/M/c로 모델링되는 서버 클래스로부터 유래되기 때문에 스케줄러에서의 지체 시간은 무시될 수 있다. 서버 클래스의 M/M/c 모델에서 평균 실행 시간은 실행 시간 밀도 함수 혹은 분포 함수를 통해서 얻을 수 있다. M/M/c에서의 실행 시간 밀도 함수 $w(t)$는 다음과 같이 알려져 있다[13].

$$w(t)=\frac{\mu e^{-\mu t}[\lambda-c\mu+\mu W_q(0)]-[1-W_q(0)][\lambda-c\mu]\,\mu e^{-(c\mu-\lambda)t}}{\lambda-c\mu+\mu},\quad t\geq 0$$

여기서 $W_q(0)$는 서버의 대기 행렬에서 기다리는 시간이 0일 확률이고, p_0는 대기 행렬 시스템에 존재하는 클라이언트 수가 0일 때의 확률이다.

$$W_q(0)=1-\frac{c\left(\frac{\lambda}{\mu}\right)^c}{c!\left(c-\frac{\lambda}{\mu}\right)}p_0$$

$$p_0=\left[\sum_{n=0}^{c-1}\frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n+\frac{1}{c!}\left(\frac{\lambda}{\mu}\right)^c\left(\frac{c\mu}{c\mu-\lambda}\right)\right]^{-1}$$

그러므로, M/M/c에서 실행 시간 분포 함수 $W(t)$는

$$W(t)=\int_0^t w(t)dt$$

$$=1-\frac{[\lambda-c\mu+\mu W_q(0)]\,e^{-\mu t}+\mu[1-W_q(0)]\,e^{-(c\mu-\lambda)t}}{\lambda-c\mu+\mu}\qquad(2)$$

이다. 부등식 $W(t) \geq p$는 주어진 c, μ, λ아래에서 사용자들의 모든 서비스 요청들 중에서 최소한 p확률 만큼은 시간 t안에 서비스가 제공될 수 있음을 의미한다. 이때 확률 p의 값을 유효 수준(Confidence Level)이라 한다. 확률 분포 함수의 성질로부터 $W(t) \geq p$의 해는 $t \geq \alpha > 0$로 기대되며, 이는 만약 시간 t를 α보다 크거나 같도록 설정한다면 최소한 p확률의 요청들은 M/M/c에서 시간 t 안에 서비스를 받을 수 있음을 보장한다. 여기서 M/M/c는 임계 액티비티에 대한 서버 클래스를 모델링한 것이다.

식 (2)로부터 $W(t) \geq p$는

$$1 - \frac{[\lambda - c\mu + \mu W_q(0)]\, e^{-\mu t} + \mu[1 - W_q(0)]\, e^{-(c\mu - \lambda)t}}{\lambda - c\mu + \mu} \geq p \qquad (3)$$

이다.

정리 4.1 $W(t) \geq p$는 항상 유일한 해 $t \geq \alpha > 0$을 갖는다.

증명) 부등식 (3)에서,

$\lambda - c\mu + \mu > 0$ 일 때,

$$[\lambda - c\mu + \mu W_q(0)]\, e^{-\mu t} + \mu[1 - W_q(0)]\, e^{-(c\mu - \lambda)t} \leq (1-p)[\lambda - c\mu + \mu] \qquad (4)$$

$\lambda - c\mu + \mu < 0$ 일 때,

$$[\lambda - c\mu + \mu W_q(0)]\, e^{-\mu t} + \mu[1 - W_q(0)]\, e^{-(c\mu - \lambda)t} \geq (1-p)[\lambda - c\mu + \mu] \qquad (5)$$

이다.

$A = \lambda - c\mu + \mu W_q(0)$, $B = \mu[1 - W_q(0)]$, $C = (1-p)[\lambda - c\mu + \mu]$, $a = \mu$, $b = (c\mu - \lambda)$라고 할 때, 부등식 (4)는

$$Ae^{-at} + Be^{-bt} \leq C \quad 단) t \geq 0, C > 0, A + B > 0 \qquad (6)$$

부등식 (5)는

$$Ae^{-at} + Be^{-bt} \geq C \quad 단) t \geq 0, C < 0, A + B < 0 \qquad (7)$$

이다. 부등식 (6)과 (7)에서 다음의 사실들을 알 수 있다. $B \geq 0$, $a > 0$, M/M/c의 안정 상태 조건 $\frac{\lambda}{c\mu} < 1$로부터 $b > 0$, $A + B = \lambda - c\mu + \mu \neq 0$, $C \neq 0$, $|A + B| \geq |C|$, 그리고 $(A+B) * C > 0$이다. 이들 사실로부터 부등식 (6)과 (7)은 그림 8의 그래프 a)와 b)에 각각 대응된다. 그러므로 이들 그래프의 해는 모두 $t \geq \alpha > 0$로 유일함이 증명된다.□

부등식 $W(t) \geq p$에서 시간 t는 한 액티비티에서 요망되는 서비스 완료시간(desired service completion time)을 의미한다. 예를 들면, 이러한 완료시간은 일반적으로 그 액티비티와 관련된 서버의 평균 실행 시간과 여분의 시간으로 구성되며 워크플로우 설계자에 의해서 결정된다. 그러므로 임계 액티비티의 서버 클래스 모델인 M/M/c에서 $W(t) \geq p$에 대한 해 $t \geq \alpha$는 "임계 액티비티에서 요망되는 서비스 완료 시간 t가 α보다 크거나 같도록 설정될 때 서비스 요청들 중에서 최소한 $p*100\%$의 요청들은 t시간 안에 서비스를 받을 수 있다" 라고 해석될 수 있다. 이때 M/M/c에서 병렬 채널의 개수 c는 임계 액티비티에 대한 중복 서버의 개수가 된다. 실제로 부등식 (3)에서 임계 액티비티의 요망되는 서비스 완료 시간 t와 유효 수준 p를 만족하기 위해서 c의 값을 조정함으로써 임계 액티비티의 최소 중복 서버의 개수 c를 결정할 수 있다.

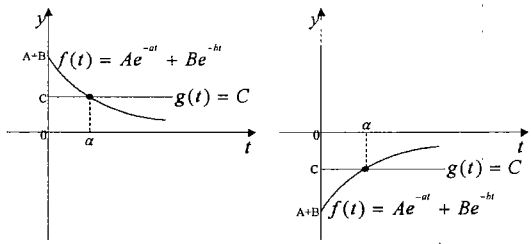


그림 8 $f(t) = Ae^{-at} + Be^{-bt}$와 $g(t) = C$의 그래프

표 1 임계 액티비티의 최소 중복 서버의 개수

서버의 서비스율	유효수준	요망되는 서비스 완료 시간	서비스 요청율	최소중복 서버의 개수
5/sec	0.9	700 msec	20/sec	6
5/sec	0.9	700 msec	30/sec	8
5/sec	0.8	1300 msec	30/sec	8
6/sec	0.9	600 msec	30/sec	7

표 1은 본 논문에서 제안된 방법을 사용하여 임계 액티비티에 대한 최소 중복 서버의 개수를 보여 주고 있고 이들 결과는 5절에서의 실험 결과들과 거의 일치한다.

5. 성능 실험

사용자들의 서비스 요청율, 액티비티 서버의 서비스율, 액티비티에서의 요망되는 서비스 완료 시간, OR과 LOOP 제어 구조에서 분기 확률 등의 정보와 3절과 4절의 방법들을 이용하여 임계 경로와 임계 액티비티에서의 최소 중복 서버의 개수를 결정할 수 있다. 이러한 분석들은 워크플로우의 전체 마감 시간 안에 대부분의 서비스 요청들을 지원할 수 있도록 보장한다. 그러나 임계 액티비티 서버의 중복은 워크플로우 관리 시스템에 과부하를 주어 비임계 액티비티들의 수행에 악영향을

미치는 경우가 발생할 수 있다. 이러한 경우는 소프트웨어적으로 해결하기 어렵기 때문에 부가적인 하드웨어 자원이 요구된다. 실제로 본 논문에서 제안된 방법을 적용할 때, 워크플로우 관리 시스템의 전체 컴퓨팅 용량을 고려할 필요가 있다.

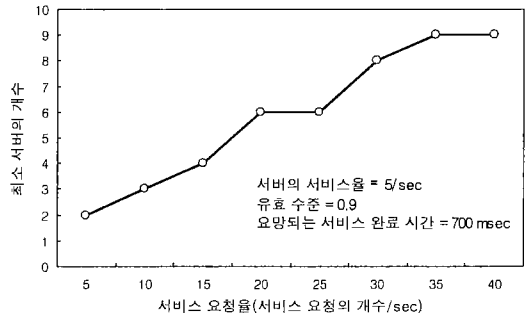


그림 9 서비스 요청율 vs. 최소 중복 서버의 개수

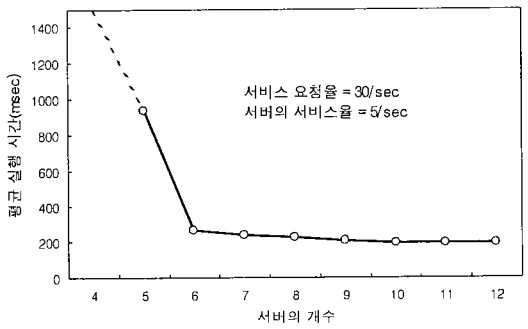
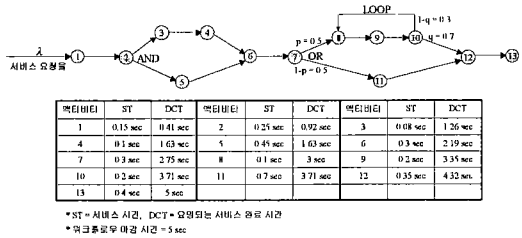


그림 10 최소 중복 서버의 개수 vs. 평균 실행 시간

임계 액티비티의 최소 중복 서버를 결정하고 시간 제약을 가지는 워크플로우의 성능을 분석하기 위해서 Systems Modeling(SM) 회사에서 개발된 Arena 시뮬레이션 소프트웨어를 이용하였다. Arena는 BPR, 전문가 시스템, 컴퓨터 시스템등을 모델링하고 성능 분석을 지원하는 시뮬레이션 도구이다. 그림 9와 10은 임계 액티비티 서버의 서비스율이 5/sec(즉, 각 서비스에 대해 20*msec*가 걸림)인 경우이다. 그림 9는 최소한 90%의 서비스 요청들이 요망되는 서비스 완료 시간 700*msec*안에 서비스를 받을 수 있도록 임계 액티비티의 최소 중복 서버의 개수를 보여 준다. 그림 10은 30/sec의 서비스 요청율일 때 중복 서버의 개수와 평균 실행 시간의 관계를 보여 준다. 중복 서버의 개수가 증가함에 따라 평균 실행 시간은 $\frac{1}{\mu}$ (즉, $\frac{1}{\text{서버의 서비스율}}$)로 수렴한다. 그림 10의 실험 경우에 평균 실행 시간은 중복 서버의 개수가 6개 까지는 급격히 감소하지만 서버의 수가 6개 이상일 때는 평균 실행 시간에는 많은 개선이 없고 워크플로우 시스템에 부하를 유발시킬 수 있다. 이는 임계 액티비티의 최소 중복 서버의 개수에 대한 분석의 필요성을 언급한다.



액티비티	ST	DCT	액티비티	ST	DCT	액티비티	ST	DCT
1	0.15 sec	0.41 sec	2	0.25 sec	0.92 sec	3	0.08 sec	1.26 sec
4	0.1 sec	1.63 sec	5	0.45 sec	1.63 sec	6	0.3 sec	2.19 sec
7	0.3 sec	2.75 sec	8	0.1 sec	3 sec	9	0.2 sec	3.35 sec
10	0.2 sec	3.71 sec	11	0.7 sec	3.71 sec	12	0.35 sec	4.32 sec
13	0.4 sec	5 sec						

* ST = 서비스 시간, DCT = 요망되는 서비스 완료 시간
* 워크플로우 마감 시간 = 5 sec

그림 11 워크플로우 스키마

그림 11은 시간 제약을 가지는 워크플로우의 성능을 평가하기 위해 정의된 워크플로우이다. 워크플로우는 13개의 액티비티들로 구성되고 워크플로우 마감 시간은 5sec이다. 각 액티비티와 관련된 서버의 서비스 시간 (즉, $\frac{1}{\text{서비스율}}$), 각 액티비티에서 요망되는 서비스 완료 시간, OR과 LOOP 제어 구조들의 분기 확률 등이 지정되어 있다. 그림 11의 LOOP 제어 구조는 3절에서 언급한 것과 같이 그림 12와 같은 순차 제어 구조로 변환된다. ICSF 알고리즘을 이용하여 얻은 임계 액티비티들은 액티비티 1, 2, 5, 6, 7, 11, 12, 13이다.

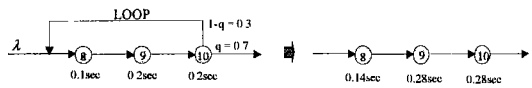


그림 12 LOOP를 순차 구조로의 변환

그림 11에서 정의된 워크플로우의 성능은 서로 다른 세가지 방법들에 의해서 평가된다. 그림 13은 각 방법에 대해 워크플로우 마감 시간을 만족하는 워크플로우 인스턴스들의 비율을 나타낸다. 첫번째 방법은 워크플로우의 액티비티들은 모두 단일 서버를 가지는 경우이다. 이를 NDS(Non-Duplication of Servers)라고 부른다. 두번째 방법은 본 논문에서 제안된 방법으로 임계 액티비

티들의 처리 용량을 늘리기 위해 최소 중복 서버를 지원하는 것으로 MNS(Minimum Number of Servers)라고 부른다. 세번째 방법은 워크플로우에 대한 분석 없이 중복 서버의 개수를 액티비티들에게 무작위로 분배하는 것으로 RNS(Random Number of Servers)라고 부른다. 여기서 RNS의 중복 서버의 개수는 MNS와 동일하다.

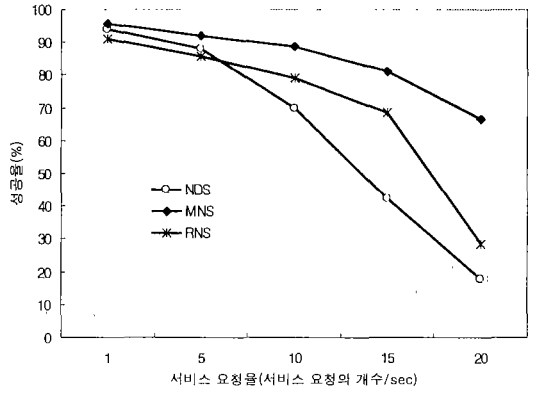


그림 13 시간 제약을 가진 워크플로우 처리의 성능 평가

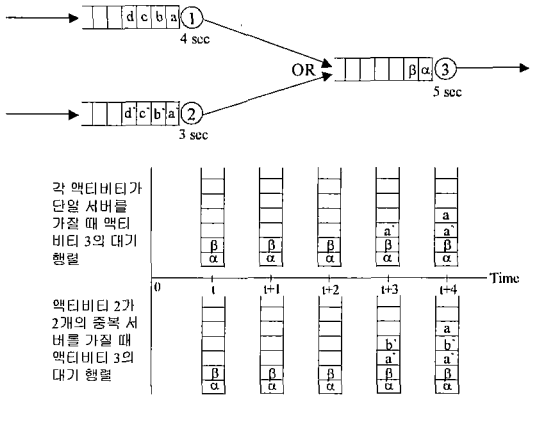


그림 14 NDS vs. RNS

NDS 방법에서 각 액티비티는 서비스 요청율을 지원할 만큼 충분한 처리 용량을 가지지 못했기 때문에 도착률이 증가함에 따라 워크플로우의 성능(즉, 워크플로우 마감 시간을 만족하는 워크플로우 인스턴스들의 비율)은 급격히 감소한다. 한편, MNS는 워크플로우 제어 흐름에 대한 분석과 임계 액티비티들에게 충분한 처리 용량을 지원하기 때문에 NDS와 RNS보다 높은 성능을 제공한다. 그림 11에서 서비스 요청율이 5보다 작은 경우에 RNS가 NDS보다 더 많은 중복 서버들을 가지고 있음에도 불구하고 RNS의 성능은 NDS보다 나쁘다는 사실은 주목할 만 하다. 그 이유는 그림 14에서 간단히 설명하고 있다. 워크플로우의 한 부분으로 액티비티 1과 2는 액티비티 3에서 OR 병합되고 각 액티비티와 관련된 서버의 서비스 시간이 명시되어 있다. 그리고 시간 t에 각 액티비티의 대기 행렬에서 기다리는 워크플로우 인스턴스들을 나타낸다. 그림 14의 아래 그림은 각 액티비티가 단일 서버(NDS의 경우)를 가질 때와 액티비티 2가 두개의 중복 서버(RNS의 경우)를 가지는 두가지 경우에 대해 시간의 변화에 따라 액티비티 3의 대기 행렬에서 기다리는 워크플로우 인스턴스들의 변화를 나타내는 대기 행렬 변화표이다. 대기 행렬 변화표에서 알 수 있듯이 액티비티 1을 지나는 모든 워크플로우 인스턴스들은 NDS보다 RNS 방법일 때 액티비티 3에서의 대기 시간이 길어진다. 그러므로 액티비티 1을 지나는 많은 워크플로우 인스턴스들은 워크플로우 전체 마감 시간을 어기게 될 확률이 그만큼 높게 된다. 이러한 현상은 워크플로우에 대한 세밀한 분석 없이 서버들을 중복하는 것은 다른 액티비티들에게 예기치 않은 과부하를 주기 때문에 워크플로우의 성능은 떨어지게 됨을 의미한다. 그러므로, 시간 제약을 가진 워크플로우의 성능을 높이기 위해서는 워크플로우 제어 흐름과 액티비티들의 처리 용량에 대한 세밀한 분석이 중요하다.

6. 결 론

워크플로우 관리 시스템은 비즈니스와 과학 분야 모두에서 응용 프로세스들을 모델링하고 제어할 수 있기 때문에 이에 대한 관심이 증가하고 있다. 비록 워크플로우는 프로세스 지향적인 사무 자동화로부터 유래되었지만 최근에는 데이타베이스와 같은 정보 자원들을 많이 사용하는 데이타 지향의 응용 분야들과 연동되고 있다. 이러한 추세에서 많은 워크플로우 응용들은 실제로 마감 시간과 같은 시간 제약 조건들을 가지고 있다.

본 논문에서는 시간 제약을 가지는 워크플로우의 성능을 개선하기 위한 기법을 소개했다. 첫번째, 우리는 워크플로우에서 가장 긴 평균 실행 시간을 가지는 임계 경로를 찾는 방법을 제안했다. 임계 경로는 워크플로우의 전체 실행 시간에 직접적으로 영향을 주기 때문에, 대부분의 워크플로우 인스턴스들이 워크플로우 마감 시간을 만족하기 위해서는 서버의 중복을 통하여 임계 액티비티들에게 충분한 처리 용량을 지원하는 것이 필요

하다. 두번째, 우리는 임계 액티비티의 처리 용량을 증가시키기 위해 최소 중복 서버의 개수를 결정하는 방법을 개발했다. 이러한 두가지 방법들을 통해 고성능 워크플로우 시스템을 지원할 수 있다. 그리고 부가적으로 본 논문에서 제안된 기법의 유용함을 보이기 위해 다양한 실험 결과들을 제공하였다.

임계 액티비티의 서버를 중복하는 것은 시간 제약을 가진 워크플로우의 성능 개선을 보장한다. 그리고 중복 서버의 개수는 액티비티에서의 요망되는 서비스 완료 시간과 밀접한 관계가 있다. 그러나, 각 액티비티에 대한 적절한 서비스 완료 시간의 선택은 워크플로우 제어 흐름과 연관이 있으므로 이를 위한 워크플로우 분석 방법이 요구된다.

참 고 문 헌

[1] P. Lawrence, Workflow Handbook 1997, John Wiley & Sons Ltd, 1997.

[2] F. Leymann and D. Roller, "Business process management with flowmark," In Proceedings of the 39th IEEE Computer Society International Conference, pp. 230-233, 1994.

[3] E. Panagos and M. Rabinovich, "Managing activity deadlines in WFMSs," In NATO Advanced Study Institute on Workflow Management Systems and Interoperability, 1997.

[4] Jin Hyun Son, Seok Kyun Oh, Myoung Ho Kim, and In Cheol Jeong, "Development of a Dynamic Scheduler supporting Multiple Servers in the Distributed Computing Environment (DCE)," Journal of KISS, vol. 6, 1998.

[5] Jin Hyun Son and Myoung Ho Kim, "On the Optimal Range of the Number of Servers in a Server Class for Distributed On-Line Requests Processing," Journal of KISS, vol. 9, 1998.

[6] H. Pozewaunig, J. Eder and W. Liebhart, "ePERT: Extending PERT for workflow management systems," In First European Symposium in ADBIS, 1997.

[7] E. Panagos and M. Rabinovich, "Predictive Workflow Management," The Third International Workshop on NGITS, 1997.

[8] E. Panagos and M. Rabinovich, "Escalations in Workflow Management Systems," Workshop on DART, 1996.

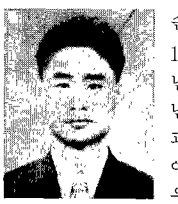
[9] Ben Kao and Hector Garcia-Molina, "Deadline Assignment in a Distributed Soft Real-Time System," In Proceedings of the 13th International Conference on Distributed Computing Systems, 1993.

[10] Ronald W. Wolff, Stochastic Modeling and the Theory of Queues, Prentice Hall, 1989.

[11] Disney, R. L., "Queucing Networks," American Mathematical Society Proceedings of Symposia in Applied Mathematics, 1981.

[12] A. Sheth, D. Georgakopoulos, S. Joosten, M. Rusinkiewicz, J. Wileden, and A. Wolf, "Report from the NSF workshop on workflow and process automation in information systems," SIGMOD Record, 1996.

[13] Donald Gross and Carl M. Harris, Fundamentals of Queueing Theory, John Wiley & Sons, 1985.



손 진 현

1996년 서강대학교 전산학과 학사. 1998년 한국과학기술원 전산학과 석사. 1998년 ~ 현재 한국과학기술원 전자전산학과 전산학 전공 박사과정 재학중. 관심분야는 데이타베이스, 미들웨어, 워크플로우, 데이타웨어하우징, 분산 컴퓨팅, CORBA

김 명 호

정보과학논문지 : 데이타베이스 제 27 권 제 1 호 참조