분산 환경에서 LTMOS의 TMO 리플리카를 이용한 실시간 로드 밸런싱의 설계 및 구현

주군호, 임보섭°, 허신, *김정국
한양대학교 컴퓨터 공학과
*한국외국어대학교 컴퓨터 및 정보통신공학부
{khjoo,bslim°,shinheu}@cse.hanyang.ac.kr, *jgkim@hufs.ac.kr

Design And Implementation Real-Time Load Balancing Using TMO Replica Of LTMOS In Distributed Environment

Koonho Joo Bosub Lim° Shin Heu *Jungguk Kim
Department of Computer Science and Engineering, Hanyang University
*Department of Computer Science and Engineering, Hankuk University of Foreign Studies

요 약

실시간 시스템이란 시간적인 제한을 가지고 작업 수행 결과의 정확도가 보장되는 시스템으로 경성 실시간 시스템과 연성 실시간 시스템으로 분류된다. 최근 분산 실시간 분야에서 새로운 패러다임으로써 폭넓게 활용되기 시작한 실시간 객체 모델인 TMO는 Kane Kim과 Kopetz에 의해 처음 제안되었다. TMO 모델은 경성 또는 연성 실시간 응용과 병렬 컴퓨팅 응용 프로그램에서 사용 될 수 있으며, 시스템의 기능적인 면과 시간 조건 수행 모두를 명확히 정의할 수 있다. TMO의 네트워크로 구성되는 실시간 분산 환경에서의 실행을 위해 몇 개의 TMO 실행 엔진이 개발 되었는데, 그 중에서 LTMOS라는 리눅스 기반의 연성 실시간 미들웨어 엔진이 한국외대 RTDCS lab.에서 개발되었다. 하지만 LTMOS의 실시간 시스템 수행 중 작업량의 과부하로 인한 deadline 위반이나, 시스템간의 분산 IPC 통신에 있어서 Channel Traffic이 빈번한 경우 실시간 시스템을 유지할 수 없다는 문제점들을 갖고 있다. 이러한 문제점들을 해결하고 조금 더 효율적인 실시간 시스템을 유지하기 위해서,. TMO 프로그램의 resource 정보를 담고 있는 ODS(Object Data Store)만을 다른 노드에 있는 자신의 TMO 프로그램 Replica로 이주해서 실시간 로드 밸런싱을 구현하는 기법을 사용하였다. 이에 본 논문에서는 TMO 프로그램들의 deadline 위반 및 Channel Traffic 부하를 감지할 수 있는 Node Monitor와 최적의 노드를 선별할 수 있는 Migration Manager를 새롭게 추가하였고, 쓰레드들의 스케줄러인 WRMT에 이주 작업을 하기 위한 부가적인 기능을 구현하였다.

1. 서 론

실시간 시스템이란 시스템 내에서 수행되는 모든 작업이 시간적인 제한을 가지고 있으며 결과의 정확도가 보장되는 시스템을 말한다. 실시간 시스템은 항공기 시스템과 원자력 발전소의 통제 시스템과 같이 예측성과 수행 시간에 대한 보장 서비스 등이 최대한 고려되어야 하는 경성 실시간 시스템과 화상 회의와 같이 시간 조건의 deadline 위반시에도 어느 정도의 복구 절차가 주어질 수 있는 연성 실시간 시스템으로 분류할 수 있다.

최근 분산 실시간 분야에서 새로운 패러다임으로써, 폭넓게 활용되기 시작한 실시간 객체 모델인 TMO(Time-triggered Message-triggered Object)는 Kane Kim과 Kopetz에 의해 처음 제안되었다[1].

TMO 개념을 적용한 실시간 분산 환경에서의 실행을 위해 몇 개의 TMO 실행 엔진이 개발 되었다. 경성 실시간 시스템 지원을 목표로 하는 DREAM Kernel이 UCI의 DREAM Lab.에서 최초로 개발 되었으며, 그 이후 윈도우와 리눅스 환경에서 TMO의 연성 실시간 수행을 지원하는 WTMOS(Windows TMO System), LTMOS(Linux TMO System)등이 미들웨어로 개발되었다[2,3,4].

이 중에서 LTMOS의 실시간 시스템 수행 중 작업량의 과부하로 인한 deadline 위반이나, 시스템간의 IPC 통신에 있어서 Traffic이 빈번한 경우 실시간 시스템을 유지할 수 없다는 문제점들을 갖고 있다. 이러한 문제점들을 해결하고, 조금 더 효율적인 실시간 시스템을 유지하기 위해서 TMO 프로그램의 resource 정보를 담고 있는 ODS(Object Data Store)만을 다른 노드에 있는 자신의 TMO 프로그램 Replica로 이주해서 실시간 로드 밸런싱을 구현하는 기법을 사용하였다.

본 논문에서는 TMO 프로그램들의 deadline 위반 및 Channel Traffic 부하를 감지할 수 있는 Node Monitor와 최적의 노드를 선별할 수 있는 Migration Manager를 새롭게 추가하고, 쓰레드들의 스케줄러인 WRMT에 이주 작업을 하기 위한 부가적인 기능을 구현함으로써 실시간 로드 밸런싱의 설계 및 구현 방법에 대하여 제시한다.

2. 관련 연구

2.1 TMO 모델

실시간 객체 모델 형태인 TMO는 정시 보장 컴퓨팅 패러다임(Timeliness guaranteed computing paradigm)을 지향하는 모델이다. 또한 TMO는 경성 실시간 응용 프로그램뿐만이 아닌 일반적인 병렬 컴퓨팅 응용 프로그램에도 사용할 수 있는 유연한 구조를 가졌으며 시스템 설계 시 정시 서비스를 보장한다.[5,6] 다음은 TMO의 특징이다.

* TMO를 구성하는 한 형태의 멤버인 시간 구동 메소드(SpM)는 객체 내의 동적 메소드로, 객체내 멤버의 특성을 갖는 스레드로 구현되어 주어진 시간 조건에 의해 자율적으로 구동된다. SpM에는 시작과 종료 시간, 주기 ,각 주기 구동시의 deadline으로 구성되는 시간 조건이 주어지고, 커널이나 TMO 엔진은 이에 대한 deadline 기반 스케줄링을 제공하여야 한다.
* TMO를 구성하는 또 다른 한 형태의 멤버인 메시지 구동 메소드는 원격 또는 국부 TMO의 메소드로부터의 IPC 메시지 수신에 의해 구동되며, 메시지에 대한 서비스 완료 시까지 deadline 스케줄링이 구현되며, 커널이나 TMO 엔진은 이에 대한 데드라인 스케줄링을 제공하여야 한다. SvM의 데드라인은 스케줄링에 반영되는 의미와 함께, SvM의 클라이언트가 데드라인 보다 더 작은 빈도로 메시지를 보내면 시간 내의 서비스가 보장되지 않음을 의미한다.
* SvM을 구동시키는 IPC 메시지는 분산 환경에도 변화 없이 그대로 적용되는, 네트워크에 투명한 분산 IPC이다. 따라서 TMO 들의 네트워크로 표현되는 실시간 시스템은 그 구성의 변화 없이 그대로 단일 노드 컴퓨팅이나 분산 컴퓨팅에 적용된다.

2.2 LTMOS

LTMOS(Linux TMO System)는 현재 주목 받고 있는 실시간 객체 모델인 TMO의 주요 기능을 C++ 객체로 구현한 분산 실시간 객체 미들웨어 엔진으로써, 리눅스 운영체제를 토대로 개발되었다. 개발 환경은 Linux 커널 2.4.X버전과 X윈도우 시스템의 GUI(Graphic User Interface)를 위해 GTK(Gimp Tool Kit)2.0 버전의 그래픽 라이브러리가 지원된다.

LTMOS는 연성 실시간 시스템으로, 기존의 리눅스 운영체제의 모든 API를 사용할 수 있으며, 리눅스 환경에서 분산 실시간 프로그래밍과 그래픽, 멀티미디어 서비스 등에 이용될 수 있는 미들웨어이다[7,8].

3. 실시간 로드 밸런싱을 위한 LTMOS 설계 및 구현

3.1 실시간 로드 밸런싱 설계

기존의 LTMOS가 제공하는 실시간 시스템에서는 작업량의 과부하로 인한 deadline 위반으로 실시간 시스템을 유지할 수 없게 된다는 점과 시스템 간의 IPC 통신에 있어서 Channel Traffic 부하로 인해 시스템 효율성이 떨어진다는 문제점들을 갖고 있다.

이러한 문제점들을 해결하고 조금 더 효율적인 실시간 시스템을 유지하기 위해서, TMO 프로그램의 resource 정보를 담고 있는 ODS(Object Data Store)만을 다른 노드에 있는 자신의 TMO 프로그램 Replica로 이주해서 실시간 로드 밸런싱을 구현하였다.

프로그램 전체가 이주하는 프로세스 이주와는 달리 ODS의 정보만을 이주함으로써 조금 더 빠르고 효율적인 로드 밸런싱을 구현 할 수 있다.

노드의 CPU 부하와 deadline 위반 및 분산 Channel Traffic 부하를 감지할 수 있는 Node Monitor와 최적의 노드를 선별할 수 있는 Migration Manager를 새롭게 추가하고, 쓰레드들의 스케줄러인 WRMT에 이주 작업을 위한 추가적인 기능을 구현함으로써 실시간 로드 밸런싱을 구현하였다.

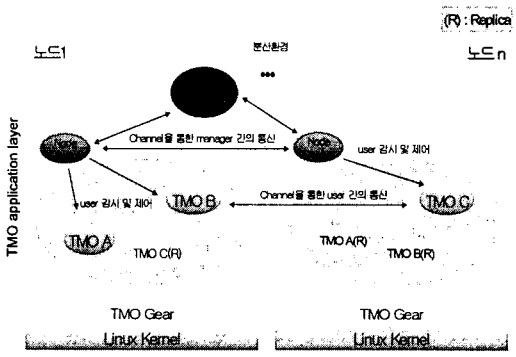

그림 1 전체 시스템 구성도

실시간 로드 밸런싱을 위한 LTMOS 기반의 시스템은 [그림1]과 같이 Node Monitor, Migration Manager, TMO 프로그램(A,B,C)들로 구성된다. [(R) - Replica]

각 부분에 대해 수행되는 역할에 대해 살펴보면 다음과 같다.

* Node Monitor : 노드내의 TMO 프로그램들을 감시하고 필요시에 ODS를 이주함.

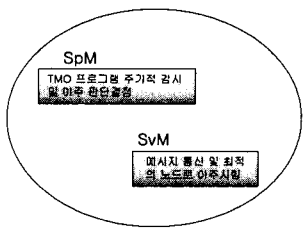

그림 2 Node Monitor 구성도

* Migration Manager : Node Monitor가 최적의 노드를 요청하면 정책에 따른 최적의 노드 제공.

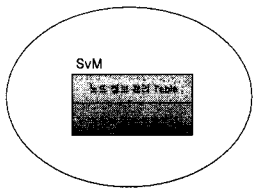

그림 3 Migration Manager 구성도

* TMO 프로그램 : SpM, SvM으로 구성되어 실제 실시간 작업을 수행함.

Node monitor와 Migration Manager의 관계를 살펴보면 다음과 같은 전체적인 흐름도가 나온다.

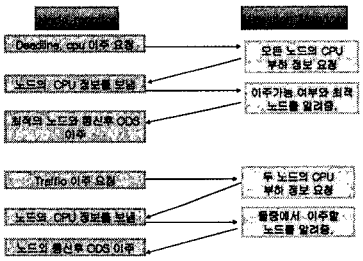

그림 4 NM 와 MM의 관계 흐름도

IPC 통신을 하는데 있어서 Channel을 통한 통신 메시지는 정해진 포맷을 통해서 구현하였으며 Channel은 Node Monitor to Migration Manager, Node Monitor to Migration Manager, Migration Manager to Node Monitor로 이루어져 있다.

3.2 구현 및 실험

LTMOS의 실시간 로드 밸런싱을 구현 실험하기 위해서 리눅스 커널 2.4.x 버전과 C++ 언어를 위한 g++ 컴파일러 및 리눅스 쓰레드를 위한 LinuxThread 라이브러리를 사용 하였다.

실험은 Deadline 위반 횟수 증가와 Channel Traffic 부하 시에 최적의 노드로 이주함으로써 개선되는 테스트를 하였다. 서로 같은 서브넷 마스크로 구성된 네트워크 상의 PC 3대에 모두 같은 TMO 프로그램(A,B,C)을 두고 시작시에 리플리카에 해당하는 프로그램의 상태만을 Sleep 상태로 두고 실제 노드에 수행중인 프로그램만을 실행한다. 한 노드의 프로그램들 중에서 한 개의 TMO 프로그램에 작업량을 늘렸을 경우 노드 부하로 인해 TMO 프로그램의 deadline 위반이 발생하게 된다. 이 때 기존의 LTMOS와 이주를 통한 LTMOS의 deadline 위반 횟수의 변화를 보여준다.

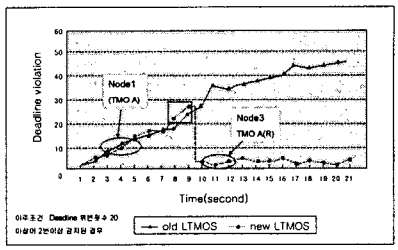

그림 5 기존 LTMOS와 개선된 LTMOS의 Deadline 변화

위 (그림 5)의 그래프 상으로 볼 때 기존 LTMOS 시스템의 경우는 부하에 따른 TMO 프로그램의 deadline 수가 지속적으로 증가해서 실시간성을 유지하지 못하는 것을 볼 수 있다. 반면에 새롭게 구현된 LTMOS 시스템의 경우는 Manager에 의한 이주를 통해서 재개한 작업의 Deadline 위반 횟수가 급격히 줄어드는 것을 볼 수 있다.

4. 결론 및 향후 연구 과제

본 논문에서는 LTMOS상의 작업 과부하와 Channel Traffic 부하로 인한 실시간성을 보장할 수 없다는 문제점을 해결하기 위해서, Node Monitor와 Migration Manager를 두고 최적의 노드로 ODS 정보만을 이주 후 재개하는 기법을 사용하여 실시간 로드 밸런싱을 구현하였다.

향후 과제로는 이주 준비 과정에 있어서 모두 Inactive 상태일 경우에만 Sleep 상태를 만드는 방법 외에 더욱 세밀한 개선책을 세우고, 신뢰성 향상을 위해 LTMOS 환경에 적합한 네트워크 프로토콜을 개발해야 할 것이다. 또한 다양한 실시간 시스템 응용을 보장할 수 있도록 지원해야 할 것이다.

참고문헌

[1] K.H. Kim and Kopetz, "A Real-Time Object Model RTO.k and an Experimental Investigation of Its Potentials," Proc. 18th IEEE Computer Software and Applications Conference, pp.392-402, November 1994.

[2] K.H. Kim, "Real-Time Simulation Techniques Based on the RTO.k Object Modeling," Proc. 20th IEEE Computer Software & Application Conference, August 1996.

[3] J.G Kim and Sang-Young Cho, "LTMOS: An Execution engine for TMO-Based Real-Time Distributed Objects," Proc. PDPTA'00 Vol. V, pp 2713-2718, Las Vegas, June, 2000.

[5] K.H Kim, "A Utopian View of Future Object-Oriented Real-time Dependable Computer System," Proc.RTCSA 94

[6] K.H. Kim, Luiz Bacellar, Yeseok Kim, Dong K. Choi. Steve Howwel and Michael Jenkins, "Ditinguishing Features and Potential Roles of the TMO Object Model," Proc. WORDS 94

[7] Kim, J.G. and Cho, S.Y., "LTMOS: An Execution engine for TMO-Based Real-Time Distributed Objects," Proc. PDPTA'00 Vol. V, pp 2713-2718, Las Vegas, June 2000.